

ROBOT EYE
A VISION BASED ROBOT SYSTEM

by

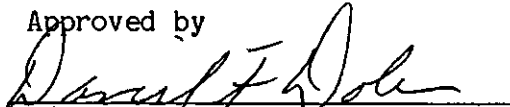
Suresh Gopaldas Advani

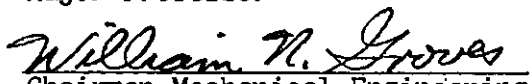
A thesis submitted to the graduate division
in partial fulfillment of the requirements
for the degree of

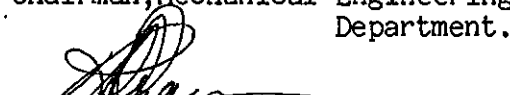
MASTER OF SCIENCE IN MECHANICAL ENGINEERING
SOUTH DAKOTA SCHOOL OF MINES AND TECHNOLOGY
RAPID CITY, SOUTH DAKOTA
1983

DEVEREAUX LIBRARY
SD SCH OF MINES & TECH
RAPID CITY, SD 57701

Approved by


Major Professor


Chairman, Mechanical Engineering
Department.


Dean, Graduate Division

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to Dr. Daniel Dolan, my thesis advisor, for introducing this field and for his help, guidance and encouragement throughout this study.

I am grateful to Dr. Ron Weger for his comments and ideas, which helped me to surmount some serious problems in Image Processing.

I am indebted to Dr. Richard Pendleton, for providing much needed moral support and encouragement throughout my stay at this school.

I would also like to acknowledge the grant provided by the SDSM&T Foundation and Caterpillar Company, which played a major role in initiating this project.

Last but not least, I would like to thank Mr. Rollo Cain, my colleague for helping me to document my thesis.

A very special thanks to my parents, to whom I dedicate this work.

ABSTRACT

ROBOT EYE is a vision based robot system, which can identify and pick up a part placed randomly in a vision controlled 'workspace' and transfer it to a predetermined location.

The hardware is comprised of a 6502-based microcomputer system, a digital camera and a 5-axis robot. The software operating system, written in BASIC, accesses machine language routines for image data acquisition, analysis and robot control. The image analysis routine calculates for a single part in the field of view, nine geometric parameters which are independent of scale, orientation and position. The robot control subroutine can use this information to grasp the part and place it in a predetermined location.

The system is user-friendly and is documented and constructed in such a manner that it can be easily used and altered for particular applications in computer vision, robot control or the combination.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
LIST OF FIGURES	v
INTRODUCTION	1
FUNCTIONAL OVERVIEW	4
SYSTEM OVERVIEW	6
VISION SUBSYSTEM	10
ROBOT SUBSYSTEM	19
SYSTEM MONITOR	28
LIMITATIONS OF ROBOT EYE	32
FUTURE WORK	34
REFERENCES	38
VITA	40
APPENDICES.	
A. DETAILS OF IMAGE ACQUISITION PROGRAM AND PROGRAM LISTING	41
B. DETAILS OF POSITION DETECTION PROGRAM AND PROGRAM LISTING	49
C. DETAILS OF ORIENTATION DETECTION PROGRAM AND PROGRAM LISTING	63
D. DETAILS OF RECOGNITION PROGRAM AND PROGRAM LISTING	74
E. MAIN PROGRAM LISTING	96

LIST OF FIGURES

	Page
1. ROBOT EYE- Hardware organization	8
2a. Te IS32 Optic RAM Chip	11
2b. Dimensions of the Optic RAM Chip	11
3. Structural Components of the Robot	20
4. Operating Envelope of the Robotarm	22
5. Robot System Flow Chart	26
6. Overview of Operational Phase of the Monitor . . .	30

INTRODUCTION

Without sensory feedback, an industrial robot cannot intelligently interact with its environment. For example, present robots require parts to be in fixed positions. Hence their use is precluded at workstations in many manufacturing activities that do not control part position. Conventionally, parts are presented to the robot by means of automatic feed systems and the robot assumes that each part is in precisely defined position.

The cost of the fixtures and feeders to accurately present the parts to the robot often exceeds the cost of the robot itself, so a means of allowing a robot to work with parts which are fed in a random manner, would save significant up-front installation costs and long term maintenance costs.

Machine Vision is one of the most important area which can advance the state-of-the-art of industrial robots, as it can provide valuable information about an environment.

This paper describes ROBOT EYE, a vision based robot system which uses a digital camera to monitor the area where the part is dispensed. The camera locates the part and verifies that it is within specifications. Then the

position and orientation coordinates are generated and converted into absolute robot position coordinates. The robot is then instructed to acquire the part and place it in a predetermined location. The process of gathering, analyzing and recognizing the object takes 3-5 seconds well within the cycle time of a typical robot.

The vision subsystem is designed to operate in a relatively known and structured environment. This is the prominent trait of an industrial vision system which is based on a pattern recognition approach, unlike general vision systems which are based on image understanding.

Industrial operations are well suited to model based analysis because of the well defined geometric descriptions associated with manufacturing items. Hence the objects were represented in terms of their geometric characteristic features such as moment of inertias, compactness ratio etc and the resulting feature vector was stored. This vector was then matched to the corresponding extracted vector during system operation.

Simplicity is important in these systems as computation time is limited. Since the ROBOTYEYE was expected to operate in real time, the processing was restricted to binary(black and white) and most of the software was implemented in machine code.

The purpose of the project was two fold:-

- 1) To build a strong foundation to enable students to pursue particular projects in machine vision, robot control or the combination.
- 2) To make it suitable for production plant use.

This paper gives an overview of ROBOTEYE followed by a description of hardware and software organization and a more detailed look at the vision, robot and monitor control software. Finally the limitations of the system are stated and the potential for future work is discussed.

FUNCTIONAL OVERVIEW

ROBOT EYE functions in four modes. A setup mode, an update mode, operational mode and a calibration mode.

Setup mode

In this mode it can either setup the path for the robot or setup the 'standard feature set' for the object.

In setting up the path, the information is to be supplied in cartesian coordinates for the approach, pickup and departure of the robot.

In the setup mode for the object, the part is shown to the camera. The 'standard feature set' is then extracted and stored in a data file. The final destination of the object should also be specified in table coordinates.

Update mode

In this mode, information about new parts can be added to the system. A procedure similar to the 'Setup mode' is followed. The part is shown to the camera and the 'Standard Feature Set' and the destination of the object added to the data file.

Operational mode

Image data acquisition and analysis is carried out for the object dispensed on the workspace. The feature set is extracted and compared to the standard feature set in order to recognize the object. Position, orientation and the final destination information is sent to the robot. The robotarm finds the part on the workspace, picks it up and transfers it to its corresponding final location. Placing of the part on the workspace is the only manual intervention otherwise the above sequence can be operated continuously.

Calibration mode

ROBOT-EYE must be calibrated when it is first setup or whenever the robot or the camera or the workspace is moved.

In this mode, the object is shown to the camera at two different locations on the table and their cartesian coordinates specified. It then works out the transformations from the vision frame to the robot frame. The calibration is simple and requires very little time.

SYSTEM OVERVIEW

Organization

ROBOT EYE is logically partitioned into independent vision, robot and monitor modules implemented on a single microcomputer. Although ROBOT EYE was implemented on a single computer, communication between subsystems was designed to allow easy substitution of new vision modules or new robots. For example, the robot could be controlled by a microprocessor independently without any major change in the organisation of the ROBOT EYE.

This particular vision subsystem reports the center of the projected area and a direction along the axis of maximum moment of inertia. Other vision modules may report altogether a different point and orientation. This can be changed in the ROBOT EYE without changing the control program.

More importantly, the workspace area can be increased or decreased by changing the exposure of the camera or its distance from the workspace. The required transformations can be obtained by using the calibration mode. This is possible because the standard feature set is independent of scale, orientation and position of the object.

Hardware

The hardware for the experimental version of ROBOT EYE as described in this paper is shown in Figure:1. The computer used is APPLE IIe. The camera is a digital camera made by MICRON TECHNOLOGY INC. The robot is a 5-axis MiniMover-5 made by FEEDBACK INCORPORATED.

Software

The software organisation for ROBOT EYE can be divided into three independent channels.

The monitor channel is responsible for controlling and coordinating the operation of the ROBOT EYE and assisting in calibration and programming of new parts.

The vision subsystem identifies the object in its 'field of view' and reports its position and orientation with reference to its frame. The monitor module performs the required transformations to the robot frame of reference. The control software at present permits only one object to be within the 'field of view' of the camera.

The vision subsystem employs just one 256x128 pixel array. The image formed is compressed into 256x64 dots which are displayed on the high resolution graphics screen of the APPLE IIe. This is achieved by reading information only from alternate pixels. The software takes care of

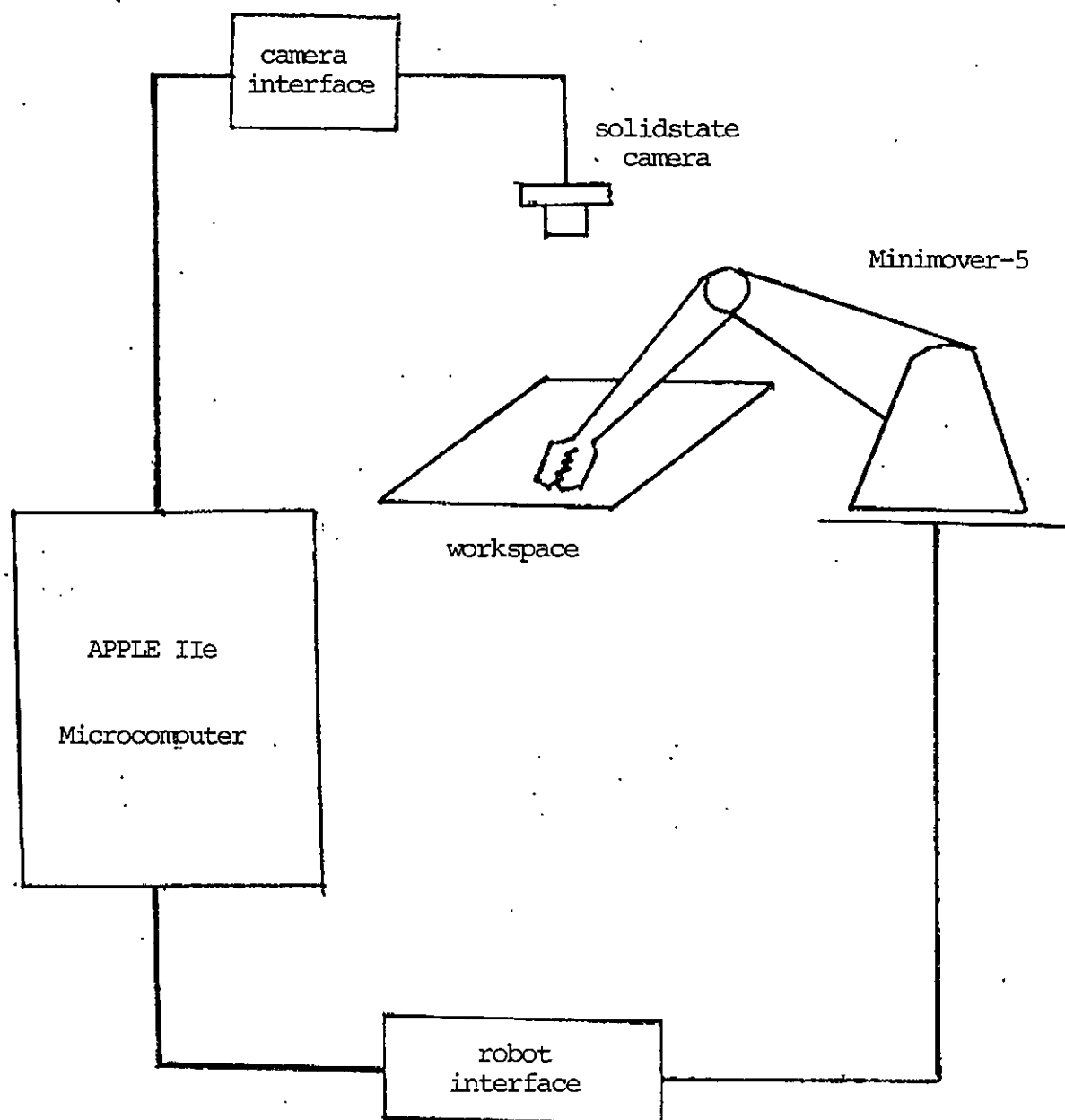


figure:1 HARDWARE ORGANIZATION

controlling the exposure of the lens.

The vision subsystem performs four main functions. Image acquisition, position detection, orientation detection and recognition.

The robot subsystem executes a previously 'taught' program to transfer the object from the workspace to a predetermined location. But it can accept information from the monitor module concerning the object's position, orientation and destination. It uses this data to 'update' the previously 'taught' program and thus carry out its required function of transferring the part.

VISION SUBSYSTEM

The vision subsystem recognizes parts within its 'field of view' and reports their position and orientation to the monitor.

Hardware

The system consists of a digital image camera which is linked to the computer by the TTL level RS-232 interface. The IS32 Optic RAM is the heart of this camera. This IS32 is a memory chip specially packaged to function as a digital image sensing device. This IS32 contains 65,536 (64K) light sensitive memory cells laid out in two planer rectangular arrays of 32,768 elements, each a matrix of 128 rows and 256 columns. The two arrays are separated by an optically nonsensitive 'dead' zone of about 25 elements wide. To avoid having a gap in the image or using complicated optical systems to eliminate it, only one of the arrays was used as an image sensor. Each of the memory elements in the matrix can be accessed randomly by simply reading in the appropriate row and column address. The Optic RAM and the dimensions of its photosensitive array are depicted in Figure: 2a and 2b.

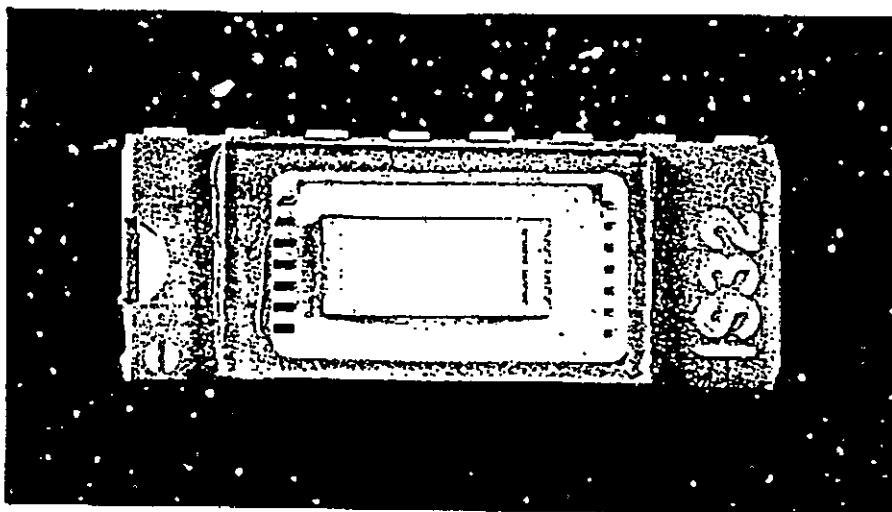


figure:2a The IS32 Optic RAM Chip

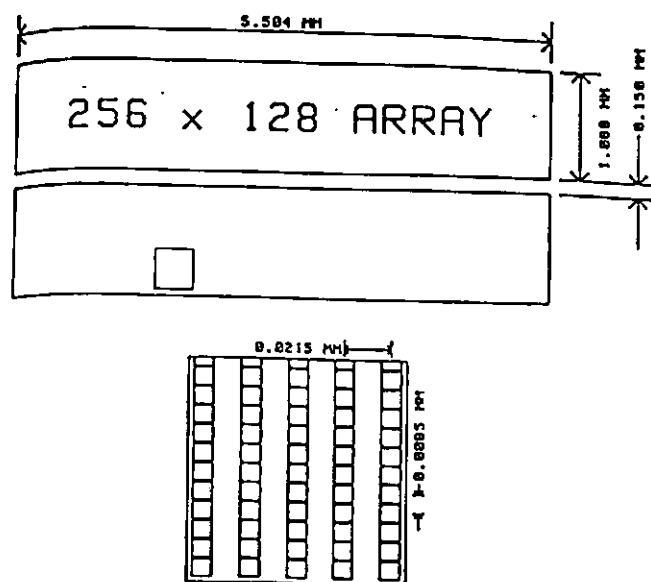


figure:2b Dimensions of the IS32 Optic RAM

Theory of Operation

An image camera built around the IS32 focuses reflected light from the viewed object and passes it through a lens onto one of the 32,768 elements array. When an individual element is struck by photons of light, the capacitor in the cell, which is initially precharged to a fixed voltage, begins to discharge towards zero volts. The capacitor discharges at a rate proportional to the light intensity throughout the duration of the exposure.

After the exposure interval has elapsed, the circuitry reads the element by addressing it as a memory cell. During the cell access, sense amplifiers within the IS32 read the capacitor's voltage value and compare it to a fixed threshold value. If the potential is above the threshold, the picture element is deemed to be black, if the potential is lower, than the picture element is declared white. The Dout pin of the Optic RAM is set to a logic 1 or 0 during the corresponding bit interval as a result of this decision

If each of these memory locations is taken and displayed on the computer's high resolution graphics screen in the same configuration, found on the surface of the Optic RAM(using a white dot to represent a 1 and a black dot to represent a 0), one will have a picture of the image focused

on the chip.

All dynamic devices require refreshing for operation. In the beginning of an image sensing cycle RS 232 circuitry addresses all the cells in the active array, filling them with positive voltages that represent logic 1s. The exposure begins with the receipt of a command from the software, which is equivalent to opening of a shutter. Then, after the appropriate exposure interval has elapsed, the control circuitry issues the refresh command, which freezes the state of the memory cells. Then the control circuitry activates the interrupt state, during which the values from these cells is transmitted. After all the bits in the array have been transmitted, then the RS-232 causes all the cells to be set to 1 again and the image sensing cycle starts over.

How fast the camera can scan the image varies according to the light intensity. The faster the elements are scanned, greater the light intensity required. At maximum speed(clock frequency 153, 600Hz),it can scan 15 frames a second.

The Interface

The RS-232 forms the interface to the expansion bus of the APPLE IIe. It owes its simplicity to the predecoding of the I/O(input-output) slot address already provided on

the APPLE's main circuit board. The transceiver buffers the TTL(Transistor-Transistor Logic) level serial data into and out of the MC6850 ACIA(Asynchronous Communication Interface Adapter). The serial bit rate is controlled by the external clock output from the drive electronics.

The 6850 ACIA comprises a data register and a status register. One can configure operating parameters (such as parity, stop bits, start bits, clocking etc.) by writing values into the status register. Before the host computer can access the link, the ACIA has to be initialised to the proper configuration. The control software does this by writing two bytes, a hexadecimal 03 followed by a hexadecimal 14, into the status register. Reading the status register allows the control program to determine when new data has been received and when ACIA is ready to send data.

In the APPLE IIe, the hexadecimal addresses of the type COnE access the status register of the ACIA on an interface card plugged into the corresponding slot, while COnF accesses the ACIA data register. The n is the hexadecimal value of the slot number plus 8. For example, suppose the interface card was plugged into slot 4, $4+8$ equals C, so address COCE will access the status register and COCF the data register.

Software

This is very vital part of the system because real time commands can be given to the camera, which will control the operating modes of the camera. The software developed deals with :-

- (1) Image Acquisition.
- (2) Position Detection
- (3) Orientation Detection
- (4) Recognition

The software consists basically of two parts; a short BASIC main program and a set of machine language subroutines. The basic program loads the machine language code from the DISK, interactively sets the correct I/O, slot number and exposure time and calls the machine language subroutines to display, freeze and process the image.

The machine language results are stored in specific memory locations of the computer, which can be accessed in BASIC to verify and make decisions accordingly.

Image Acquisition.

The BASIC subroutine calls the machine language routine to display the image. The machine code makes sure that high resolution graphics is being displayed. It then initializes the ACIA and sends a command which clears the

Optic RAM and tells the camera to begin the exposure. The program then waits for the duration of the exposure, which again can be controlled by the software.

The next step is to read the image from the camera and display it on the high resolution graphics screen. To save time and memory, the software sends the picture straight to high resolution graphics memory rather than reading into a separate buffer. The software selects the alternate pixel with 7-bit data words to display 128 X 256 array on 64 X 256 dots of the high resolution screen.

Before any part of the picture is received, a number of memory pointers are set up to facilitate proper communication. A command is sent to the camera to begin transmitting the image and the program loops back to read in each byte of the image and place it on the screen. The control software knows how many bytes of image data it should receive from the camera.

As the APPLE's high resolution screen is mapped non-linearly into memory space, a lengthy table in machine code provides the starting address for each consecutive row of the high resolution screen. The program gets the address of the beginning of each row and then reads the required number of bytes from the camera, placing them consecutively on the screen.

Once the image is on the screen, a command is send to the camera to refresh without sending. This gets it ready for the next exposure. Finally the software can also control the number of times you want to refresh the image before terminating this subroutine and freezing the final image in the high resolution graphics memory. For more details refer to Appendix-A.

Position Detection

The position of the image is specified with reference to the high resolution screen coordinates. The BASIC program calls the machine language subroutine which scans the memory where the image data was stored to calculate the centroid of the image. The centroid is expressed in (x,y) , where x represents a particular row and y represents a particular column. Refer to Appendix-B for further details.

Orientation Detection

The machine language subroutine scans the image data and calculates the Products of Inertia and the Principle Moments of Inertia, and returns the control to the BASIC, where the orientation of the major axis is determined with respect to a row on the screen. This is only one way to determine the orientation. But orientation with respect to a column or with respect to a minor axis can also be

determined by changing just the BASIC program. (Refer to Appendix-C)

Recognition

The software was developed so that ROBOT EYE can recognize parts regardless of scale, position or orientation.

The machine language can process the image very fast and calculate up to the third moments of inertia, determine the perimeter and area of the image. Once the control is returned to BASIC, the independence of scale is accomplished by scaling these moments with respect to the area. Then it uses a set of geometric relationships to obtain nine parameters independent of scale, position and orientation(refer to appendix-D).

Thus the software refines and reduces the image data to basic characteristics called as the 'Feature Set' needed to evaluate and identify the original object. The 'Feature Set' is then compared with a reference 'Standard Set' stored in the system.

ROBOT SUBSYSTEM

The robot subsystem is comprised of a MiniMover-5 manipulator, a five jointed mechanical arm, which is driven by digital commands from the APPLE IIe.

Physical Characteristics

The major structural components are shown in Figure: 3. The manipulator consists of a fixed base within which is housed the computer interface card. From the rear of the base extends the interface cable and the D.C power cord. The body swivels relative to the base on a hollow shaft which is attached to the base. This is known as the base joint. Six stepper motors with their corresponding gear reductions, are mounted on the body and control each of the joints.

At the upper end of the body is attached the upper arm. The upper arm rotates relative to the body on a shaft. This is known as the shoulder joint. Similarly, the forearm is attached to the upper arm by another shaft. This is known as the elbow joint. Finally the gripper(hand) is attached to the forearm at the wrist joint.

The end of the hand can be positioned anywhere within

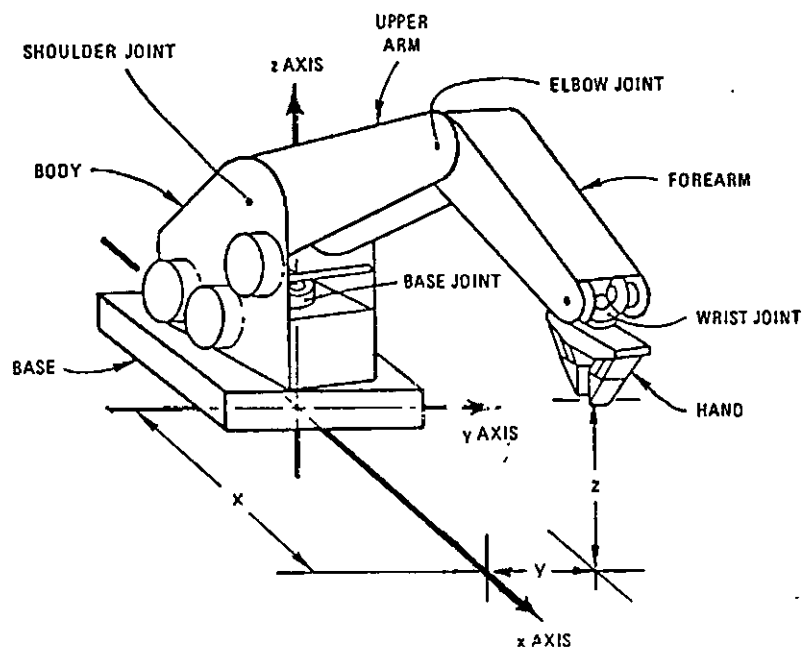


figure:3 Structural Components of the Robot

a partial sphere which has a radius of 17.5 inches as shown in Figure: 4. The maximum speed of the robot is limited to 2 to 6 inches per second, depending upon the weight of the object being handled. The MiniMover-5's hand has a maximum opening of 3 inches. The 2-bar linkages maintain the fingers parallel during the opening and closing. A builtin sensor permits adaptive control so that a presence of a object may be detected.

Software

Robot programming languages are generally classified by non-textual programming and textual programming methods. In non-textual programming, the robot motions are taught by the user using the teach pendant, and each move is recorded in the memory for ready playback. For this reason non-textual programming is also referred to as 'teaching by showing'. The main problem with this mode of programming is that it is difficult and sometimes impossible to incorporate sensory information if the sensory data are not available until actual robot motion. Hence to remedy this deficiency, the textual mode of programming is essential. The MiniMover-5 can be programmed into two levels of textual mode:-

- (1) Joint level
- (2) Manipulator level

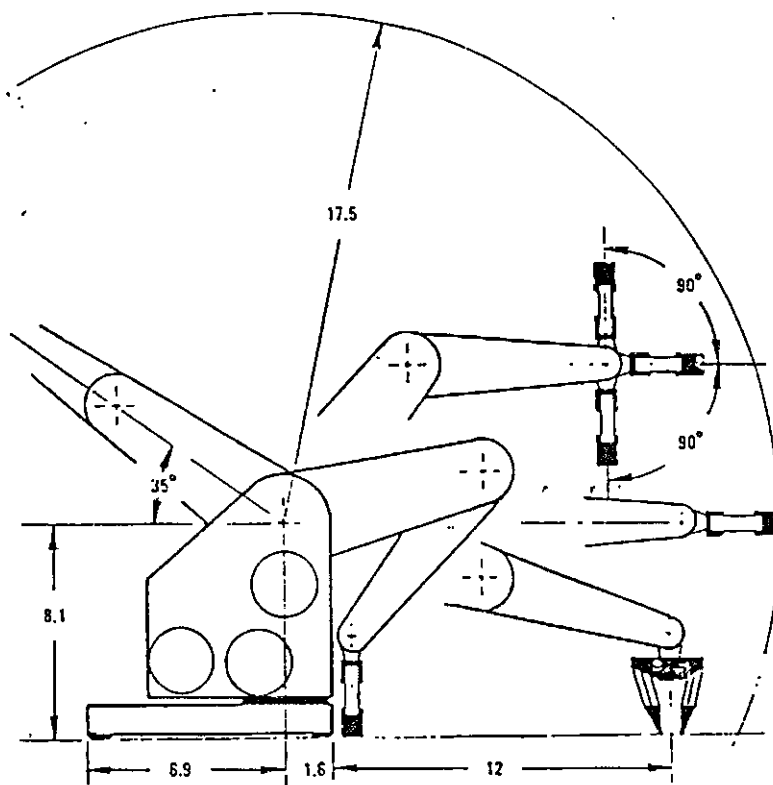


figure:4 Operating Envelope of the Robotarm

Joint Level Languages

In joint level programming, the required manipulator motion to accomplish a task is specified by a sequence of joint motion movements. The instructions are of the following type:-

STEP J1,J2,J3,J4,J5,J6,D

where J1,J2.....J6 represent joint angles. The sign of each number indicates the direction each motor should be driven. The magnitude indicates the number of steps. The numbers J1 to J6 with the directions for the positive numbers are as follows:-

J1: Base swivel - clockwise

J2: Shoulder bend - downwards

J3: Elbow bend - upwards

J4: Right wrist - upwards

J5: Left wrist - upwards

J6: Hand open

The J4,J5 and J6 are also called as the 'Roll, Pitch and Yaw' movements. The speed is determined by D.

This type of programming is undesirable because the coordinates natural to the human are cartesian, cylindreical or spherical. This transformation can be accomplished by the manipulator level languages.

Manipulator Level Languages

Manipulator level programming allows the user to program a task in terms of sequence of robot motions represented by the end effector(gripper) locations. The motion statements programmed by the user are of the following type:-

STEP X,Y,Z,L1,L2,L3,D

where X,Y,Z represent the position of the end effector and L1,L2,L3 represent the orientation(roll, pitch and yaw), all with respect to global coordinates.

The ROBOTEEYE software can compute the transformations from cartesian to joint coordinates by solving the kinematic equations for the MiniMover-5.

Initialisation Requirements

Before a program is executed, the arm should be moved to a known initial 'HOME' position. This is necessary in order to match the position of the joints to the position assumed by the program. It is not necessary to initialize everytime because the robotarm is made to return to the 'HOME' position after the completion of the cycle in the ROBOTEEYE. Hence initialition of the 'HOME' position is required only if the system is moved or interrupted.

Operation

The robot programming subsystem is implemented as two independent tasks. One task is required for robot program development. The second deals with the execution.

Development Mode

In this mode the robot is taught the path. The approach, pickup and departure information is supplied in terms of X, Y and Z coordinates of the table. This information can be different for different objects.

Execution Mode

In this mode the robot will follow the same path which was previously taught, for that particular object. I can accept information concerning the object, its location, orientation and destination from the vision subsystem and use it to 'update' its assigned path. This information is provided by the monitor module, which transforms the screen coordinates into actual cartesian coordinates. The flow-chart in Figure: 5 depicts the function of robot subsystem in the execution mode.

Thus the robot subsystem software along with the monitor modules helps to keep track of the operating

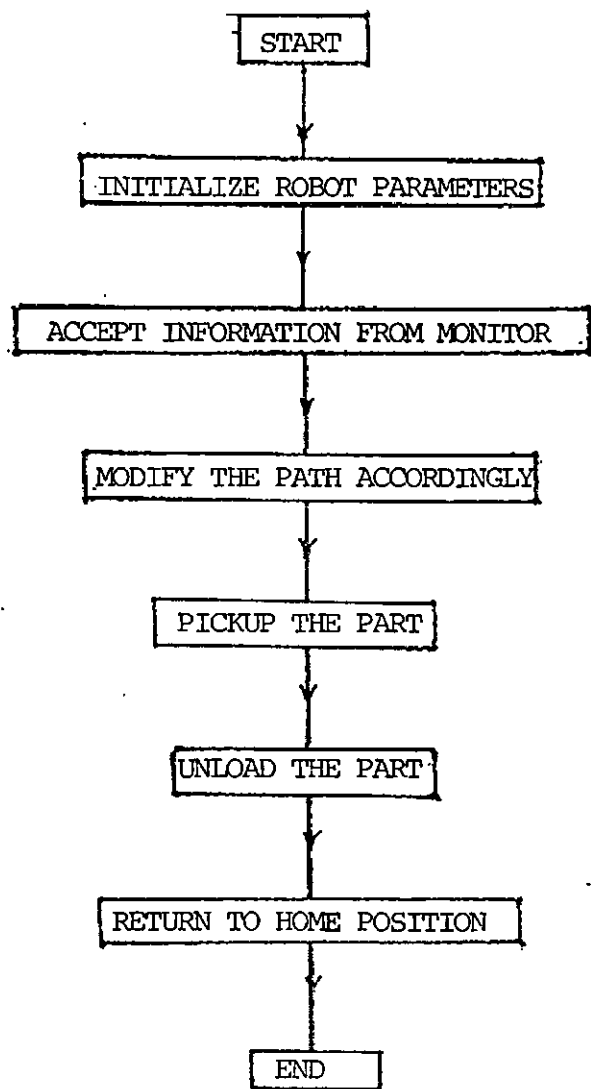


figure:5 Robot System Flow Chart

envelope of the MiniMover-5. If the monitor module makes an unconventional request to reach a point behind the scope of the arm, the program will terminate and return the robot to the 'HOME' position. Hence all these means in the robot subsystem provide practically all the capabilities to the ROBOT EYE without spending a major effort in developing a powerful robot language.

SYSTEM MONITOR

The monitor coordinates the operation of the vision and the robot subsystems, while calibration and part programming as well as during the operational phase. As stated earlier, calibration is required during initial setup or whenever the camera or the robot or the workspace is moved. Part programming is required only during the initial setup and whenever ROBOT EYE is modified to handle a new part.

Calibration

In ROBOT EYE, the part position determined by the vision subsystem defines the position and orientation relative to the screen frame. The approach, pickup and departure are all programmed with respect to the table frame. Hence, calibration is the process whereby the relationship between the screen coordinates(Vision coordinate system) and the table coordinates(Robot coordinate system) is determined.

In this mode the object should be viewed by the camera at two different locations on the table. The table coordinates, where the object was placed should also be specified. The monitor system can then compute the

relationship between the vision and robot frames and store it into a data file. Determining the coordinate transformation completes the calibration.

Operational Phase

In this phase the monitor controls and coordinates the operation of the ROBOT EYE. The three important functions it carries out are:-

- (1) It acquires the 'Feature Set' from the vision subsystem and compares it with the 'Standard Feature Set' in order to identify the object.
- (2) It transforms the screen coordinates of the position and orientation of the object to the table frame of reference.
- (3) It converts the path of the robot from cartesian coordinates into number of motor steps. It feeds this information to the robot subsystem for execution.

Figure: 6 illustrates the overall logic of monitor. Although the logic is straightforward, the monitor deals with two rather subtle problems.

The first involves modification of the orientation of the gripper. The robot has limited range of motion in all its joints. The gripper has limited orientation. Since this gripper is symmetric about 180 degrees, the monitor can effectively achieve the same orientation by rotating through

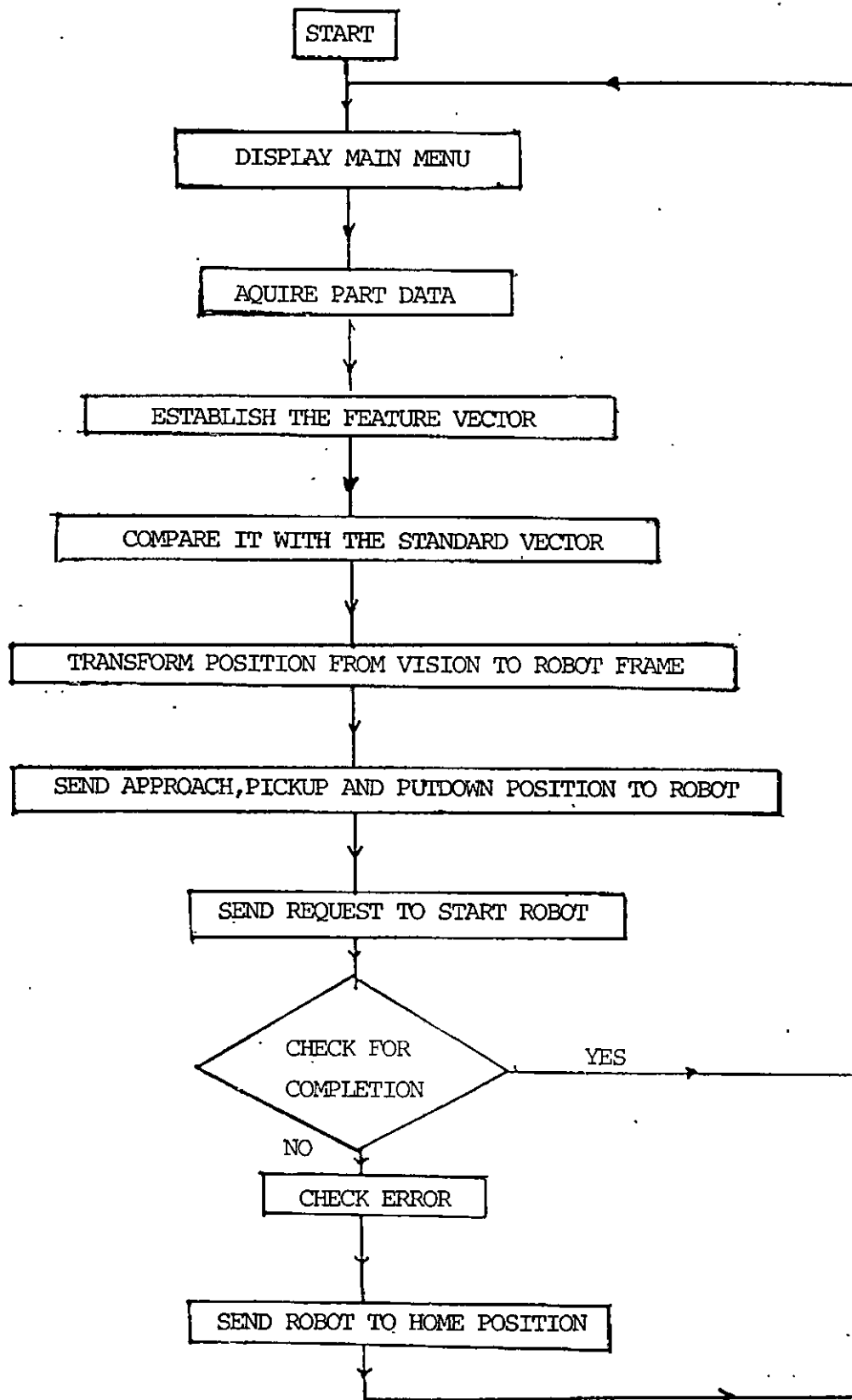


figure:6 Overview of Operational Phase of System Monitor.

the smallest angle for a particular pickup position.

Error recovery is the second problem. As mentioned earlier, the possibilities of parts out of reach, impossible robot positions etc. increase significantly when the robot's path is being changed for every cycle. Errors detected by either the vision or robot subsystem are reported to the monitor. The monitor then takes action, by returning the robot to the 'HOME' position and returning the control to the main menu.

LIMITATIONS OF ROBOT EYE

The system described above is capable of only binary level processing. Although more information about the object can be extracted by gray level processing, the computation time involved will be much more, and hence only 'black and white' type of analysis was used. As a result, processing of the image could be done only for geometric analysis in 2-D plane. It cannot be utilized for analysis of objects which require that surface characteristics be quantified.

The ROBOT EYE, after acquiring the image does not carry out any filtering as it assumes that the operation will be in a relatively known and structured environment. Thus any noise in the image will lead to wrong interpretations. Therefore the contrast of the object against its background should be greater than the local lighting variations around the feature of interest. Usually lighting variations are caused by point light sources or by interferences from ambient light. Hence some kind of structured lighting is essential. But the vision subsystem does not require any complicated lighting arrangements or other impractical ways of enhancing contrast.

The workspace is controlled by the exposure of the

camera and its distance from the workspace. Thus the size of the object which can be processed by the system is limited by the 'field of view'.

The calibration procedure requires the plane of the camera to be parallel to the workspace in order to compute the correct transformations.

The ROBOT EYE can process only one object at a time and moreover the object should be free from discontinuities.

The precision of the system is limited by the accuracy of the transformation of the vision subsystem to robot subsystem coordinates and by the precision of the robot movements.

FUTURE WORK

ROBOT EYE is now a strong foundation for development of many educational and practical applications in computer vision or robot control.

Some of the projects which can be further developed are

- 1) Image processing can be extended to analyse more than one object in its field of view. This can be easily implemented by undergoing minor changes in the 'row' and 'column' machine language routines.
- 2) More information about the object can be extracted if the object is viewed from two different angles. This would require an additional digital camera. The basic processing of the image can be done by using the same control program. Further development can be achieved by deriving a relationship between the 'Feature Sets' which were extracted from two different views.
- 3) Programs could be developed to break up a complex image into two or three different blobs (groups of connected pixels in a binary image). Then each blob can be processed independently for a 'Feature Set' using the ROBOT EYE control software. This development will enable the system to process complex objects.
- 4) The present vision subsystem can be tested under different types of structured lighting. This could result in a lighting arrangement, which is simple yet produces

a resonable contrast with minimum noise and minimum distortion of the image.

- 5) The possibility of distributing the vision subsystem and the robot subsystem functions among different computers should be pursued. This would not only decrease the cycle time but would also allow the user to process the image for more parameters in real time.
- 6) Positions to which the robot moves can be programmed in non-textual languages when in the development mode and then a reverse transformation to obtain the cartesian coordinates can be accomplished. This would enhance the accuracy while calibration, as the object can be shown to the vision subsystem and instead of specifying the coordinates the user can manually move the robot to the object's position. The software can then obtain the conversion from the joints to cartesian coordinates.

Some advanced level programming subroutines that can be developed on this system are:

- 1) To generate a gray scale level. This would aid complex part recognition or for the analysis of surface characteristics. This technique of processing has a lot of potential for handling scenes with clustered background and uncontrolled lighting.
- 2) Software to develop a Fast Fourier Transform of the image can be developed, which would give more than

enough information to recognize an object.

- 3) Task level programming, as an attempt to develop a robot language, should be pursued. Here the user specifies the task in terms of statements describing object manipulation, not robot motion. The task level programming is very attractive in the programmer's point of view; instructions are easy to read and write. This is particularly important to manufacturing environment since most machine operators do not have programming experience.

Practical Applications

ROBOT EYE could be further developed to pick up parts from a moving conveyor belt. This would require a few additions to the main program. The conveyor belt could be halted under the camera station where it will serve as the workspace. Once the image is acquired, the belt could be moved again and by the time it reaches the robot station the image should be processed. The position and orientation information should be supplied to the robot subsystem and the robot can track the part on the conveyor and pick it up.

The addition to the system would be to instrument the belt with a position and speed detection device, and workout appropriate translation transformations in the calibration mode. Once the system is setup, it can be used for sorting of parts or for transferring of parts. This sequence can

operate continuously without manual intervention except for placing of parts on the conveyor.

With all these developments, the system will not be far from being compatible to find applications in the industry.

REFERENCES

- 1) M. C. Leu, "Developments in Robotics Software Systems", Computer Integrated Manufacturing, Boston, Massachusetts, Nov, 1983.
- 2) W. B. Gevarter, "Machine Vision: A report on the state of the art", CIME, April, 1983.
- 3) Dr. Chris Wieland, "The Microneye", BYTE, Oct, 1983.
- 4) Steve Ciarcia, "Build the Micron D-Cam Solid State Video Camera", BYTE, Oct, 1983.
- 5) R. Jain and Susan Hayes, "Imprecision in Computer Vision", COMPUTER, August 1982.
- 6) D. L. Waltz, "A Parallel Model for Low Level Vision", Computer Vision Systems, Academic Press, Inc, 1978.
- 7) W. J. Clark, W. N. Frick, "Robot Loading of a P.C. Board Test Systems", Applied Machine Vision Conference, Memphis, Tennessee, Feb, 1983.
- 8) S. W. Holland, L. Rossol, M. R. Ward, "Consight-I, A Vision Controlled Robot System For transferring Parts from Belt Conveyors", Computer Vision and Sensor-Based Robots, Plenum Press, 1979.
- 9) E. L. Hall, "Computer Image Processing", Academic Press, 1979.
- 10) R. C. Gonzalez, P. Wintz, "Digital Image Processing", Addison Wesley Publishing Company, 1979.
- 11) Brady, Hollerbach, Johnson, Mason, "Robot Motion: Planning and Control", MIT Press, 1982.
- 12) Paul Richard P., "Robot Manipulators: Mathematics, Programming and Controls", MIT Press, 1982.
- 13) J. Stanton, "Apple Graphics and Arcade Game Design", The Book Company, 1982.
- 14) R. K. Miller, "Machine Vision for Robotics and Automated Inspection Volume-I, II", Technical Insights, Inc., New Jersey, 1982.
- 15) Don Inman, Kurt Inman, "Apple Machine Language", Prentice Hall, 1982.
- 16) "Disk Operating Manual", Apple Computers Inc., 1981.

- 17) "Applesoft Reference Manual" , Apple Computers Inc., 1981.
- 18) "Microneye Operation Manual" , Micron Technology Inc., 1981
- 19) "Robotics Reference and Application Manual" , Feedback Inc., 1981

VITA

Suresh Gopaldas Advani was born on 19 August, 1959 in Pune, INDIA. He graduated from St. Vincent's High School, Pune in May 1976. He completed all the requirements of Bachelor of Technology degree in Mechanical Engineering at INDIAN INSTITUTE OF TECHNOLOGY, Bombay, INDIA in May 1982. He worked as a trainee engineer in ATLAS COPCO LTD, INDIA for 3 months after his graduation.

Mr. Advani was a teaching assistant at SOUTH DAKOTA SCHOOL OF MINES AND TECHNOLOGY, Rapid City, South Dakota from August 1982 to May 1983 and from September 1983 to December 1983. He also served as a research assistant from June 1983 to August 1983.

Mr. Suresh Advani is a member of IIT Alumni Association, India and American Society of Mechanical Engineers.

APPENDIX-A

This appendix deals with 'Image Acquisition'. Machine language subroutine 'CAMASM' loads the high resolution graphics screen with the picture of the object and freezes it in the high resolution graphics memory.

A brief description of this program, along with a flow chart, memory map and program listing is given.

ASSEMBLY ROUTINE "CAMASM"

This is a machine language routine which allows input from the camera to be displayed in High Resolution Graphics Page 2.

Some of the important subroutines of this program are:

1) JSR 8108 *

- * This subroutine checks for the slot number in
- * memory location \$303 and accordingly takes input
- * from the camera

2) JSR 811A *

- * This subroutine clears the High Resolution
- * Graphics Page 2.

3) JSR 8132 *

- * Sets the soft switches to display the High
- * Resolution Graphics Page 2.

4) JSR 813F *

- * Takes the input from the camera and converts it
- * into an ON or OFF bit and stores that
- * information in the HGR2 memory

5) JSR 8153 *

- * Loads the exposure time into memory
- * location \$301(MSB) and \$302(LSB) and then goes
- * through JSR 8177 until exposure time is reached.

6) JSR 8177 *

- * Delay loop which exposes the camera to the object
- * for 0 to 255 counts for each increment
- * of exposure time

MEMORY MAP

- \$300 * Stores the decimal command(CM). CM is a function of
 - * no. of pixels, pixel width and no. of arrays used.
- \$301 * Stores the MSB of Exposure Time.
- \$302 * Stores the LSB of Exposure Time
- \$303 * (Slot # of the camera)*16
- \$304 * Stores the input Key Board for exit
- \$305 * Stores input from Key Board to control the Microneye.
- \$306 * LSB to store the light level.
- \$307 * MSB to store the light level.
- \$308 * Number of bytes that are required in a row. ?
- \$309 * ^{LSB}MSB of no. of rows to be used. 00
- \$30A * ^{MSB}LSB of no. of rows to be used. 01
- \$30B * Counter for no. of scans of the camera.
- \$30C * Number of bytes to be used in a row. ?
- ~~\$822B~~ * Stores the first memory location of every row of the
 - * HGR Page as the memory is not organised.
- \$C000 * Allows input from the Key Board even while executing.
- \$C0C0- * Input from slot 4 ,if the camera is located there.
- \$4000- * High Resolution Graphics memory
- \$8400 * Only way to get out of the camera mode with the image
 - * still loaded in HGR2 memory. It comes out of the
 - * camera mode when the value is equal to 5.

PROGRAM LISTING OF CAMASM

8000-	4C 0F 80	JMP	\$800F	*
8003-	4C E9 81	JMP	\$81E9	*
8006-	4C 81 81	JMP	\$8181	*
8009-	4C AF 81	JMP	\$81AF	*
800C-	4C DF 81	JMP	\$81DF	*
800F-	AD 2A 82	LDA	\$822A	*
8012-	F0 06	BEQ	\$801A	*
8014-	CE 2A 82	DEC	\$822A	*
8017-	20 1A 81	JSR	\$811A	*
801A-	20 32 81	JSR	\$8132	*
801D-	20 08 81	JSR	\$8108	*
8020-	EE 00 84	INC	\$8400	*
8023-	A9 05	LDA	#\$05	*
8025-	CD 00 84	CMP	\$8400	*
8028-	D0 03	BNE	\$802D	*
802A-	4C 01 81	JMP	\$8101	*
802D-	AD 0B 03	LDA	\$030B	*
8030-	F0 16	BEQ	\$8048	*
8032-	AD 00 03	LDA	\$0300	*
8035-	09 13	ORA	#\$13	*
8037-	85 1C	STA	\$1C	*
8039-	20 3F 81	JSR	\$813F	*
803C-	20 53 81	JSR	\$8153	*
803F-	AD 00 03	LDA	\$0300	*
8042-	29 FC	AND	#\$FC	*
8044-	85 1C	STA	\$1C	*
8046-	D0 08	BNE	\$8050	*
8048-	20 53 81	JSR	\$8153	*
804B-	AD 00 03	LDA	\$0300	*
804E-	85 1C	STA	\$1C	*
8050-	20 3F 81	JSR	\$813F	*
8053-	A2 00	LDX	#\$00	*
8055-	8E 06 03	STX	\$0306	*
8058-	8E 07 03	STX	\$0307	*
805B-	8E 1C 82	STX	\$821C	*
805E-	A0 00	LDY	#\$00	*
8060-	BD 2B 82	LDA	\$822B,X	*
8063-	85 06	STA	\$06	*
8065-	E8	INX		*
8066-	BD 2B 82	LDA	\$822B,X	*
8069-	85 07	STA	\$07	*
806B-	E8	INX		*
806C-	84 1B	STY	\$1B	*
806E-	AC 03 03	LDY	\$0303	*
8071-	B9 8E C0	LDA	\$C08E,Y	*
8074-	4A	LSR		*
8075-	B0 21	BCS	\$8098	*
8077-	A9 00	LDA	#\$00	*
8079-	85 19	STA	\$19	*

807B-	A9 15	LDA	#\$15	*
807D-	85 1A	STA	\$1A	*
807F-	C6 19	DEC	\$19	*
8081-	D0 0F	BNE	\$8092	*
8083-	C6 1A	DEC	\$1A	*
8085-	D0 0B	BNE	\$8092	*
8087-	AD 30 C0	LDA	\$C030	*
808A-	AD 00 C0	LDA	\$C000	*
808D-	30 47	BMI	\$80D6	*
808F-	4C 1D 80	JMP	\$801D	*
8092-	B9 8E C0	LDA	\$C08E,Y	*
8095-	4A	LSR		*
8096-	90 E7	BCC	\$807F	*
8098-	B9 8F C0	LDA	\$C08F,Y	*
809B-	A4 1B	LDY	\$1B	*
809D-	4A	LSR		*
809E-	90 08	BCC	\$80A8	*
80A0-	EE 06 03	INC	\$0306	*
80A3-	D0 03	BNE	\$80A8	*
80A5-	EE 07 03	INC	\$0307	*
80A8-	2A	ROL		*
80A9-	C0 28	CPY	#\$28	*
80AB-	B0 02	BCS	\$80AF	*
80AD-	91 06	STA	(\$06),Y	*
80AF-	C8	INY		*
80B0-	CC 08 03	CPY	\$0308	*
80B3-	D0 B7	BNE	\$806C	*
80B5-	EC 09 03	CPX	\$0309	*
80B8-	D0 A4	BNE	\$805E	*
80BA-	E0 00	CPX	#\$00	*
80BC-	D0 0B	BNE	\$80C9	*
80BE-	EE 1C 82	INC	\$821C	*
80C1-	AD 1C 82	LDA	\$821C	*
80C4-	CD 0A 03	CMP	\$030A	*
80C7-	D0 95	BNE	\$805E	*
80C9-	AD 0B 03	LDA	\$030B	*
80CC-	D0 08	BNE	\$80D6	*
80CE-	AD 00 C0	LDA	\$C000	*
80D1-	30 03	BMI	\$80D6	*
80D3-	4C 1D 80	JMP	\$801D	*
80D6-	AD 00 03	LDA	\$0300	*
80D9-	29 FC	AND	#\$FC	*
80DB-	09 11	ORA	#\$11	*
80DD-	85 1C	STA	\$1C	*
80DF-	20 3F 81	JSR	\$813F	*
80E2-	A9 20	LDA	#\$20	*
80E4-	8D 05 03	STA	\$0305	*
80E7-	A9 00	LDA	#\$00	*
80E9-	8D 04 03	STA	\$0304	*
80EC-	AD 00 C0	LDA	\$C000	*
80EF-	10 16	BPL	\$8107	*
80F1-	2C 10 C0	BIT	\$C010	*

80F4-	EE 04 03	INC	\$0304	*
80F7-	8D 05 03	STA	\$0305	*
80FA-	C9 D1	CMP	#\$D1	*
80FC-	D0 09	BNE	\$8107	*
80FE-	20 0D 81	JSR	\$810D	*
8101-	AD 51 C0	LDA	\$C051	*
8104-	AD 54 C0	LDA	\$C054	*
8107-	60	RTS		*
8108-	A9 03	LDA	#\$03	*
810A-	84 1B	STY	\$1B	*
810C-	AC 03 03	LDY	\$0303	*
810F-	99 8E C0	STA	\$C08E,Y	*
8112-	A9 14	LDA	#\$14	*
8114-	99 8E C0	STA	\$C08E,Y	*
8117-	A4 1B	LDY	\$1B	*
8119-	60	RTS		*
811A-	A2 00	LDX	#\$00	*
811C-	A0 00	LDY	#\$00	*
811E-	84 06	STY	\$06	*
8120-	A9 40	LDA	#\$40	*
8122-	85 07	STA	\$07	*
8124-	8A	TXA		*
8125-	91 06	STA	(\$06),Y	*
8127-	C8	INY		*
8128-	D0 FB	BNE	\$8125	*
812A-	E6 07	INC	\$07	*
812C-	E8	INX		*
812D-	E0 20	CPX	#\$20	*
812F-	D0 F4	BNE	\$8125	*
8131-	60	RTS		*
8132-	AD 53 C0	LDA	\$C053	*
8135-	AD 57 C0	LDA	\$C057	*
8138-	AD 55 C0	LDA	\$C055	*
813B-	AD 50 C0	LDA	\$C050	*
813E-	60	RTS		*
813F-	84 1B	STY	\$1B	*
8141-	AC 03 03	LDY	\$0303	*
8144-	B9 8E C0	LDA	\$C08E,Y	*
8147-	29 02	AND	#\$Q2	*
8149-	F0 F9	BEQ	\$8144	*
814B-	A5 1C	LDA	\$1C	*
814D-	99 8F C0	STA	\$C08F,Y	*
8150-	A4 1B	LDY	\$1B	*
8152-	60	RTS		*
8153-	AD 02 03	LDA	\$0302	*
8156-	85 1A	STA	\$1A	*
8158-	E6 1A	INC	\$1A	*
815A-	AD 01 03	LDA	\$0301	*
815D-	85 19	STA	\$19	*
815F-	E6 19	INC	\$19	*
8161-	AD 01 03	LDA	\$0301	*
8164-	D0 05	BNE	\$816B	*

8166-	AD 02 03	LDA	\$0302	*
8169-	F0 0B	BEQ	\$8176	*
816B-	20 77 81	JSR	\$8177	*
816E-	C6 19	DEC	\$19	*
8170-	D0 F9	BNE	\$816B	*
8172-	C6 1A	DEC	\$1A	*
8174-	D0 F5	BNE	\$816B	*
8176-	60	RTS		*
8177-	84 1B	STY	\$1B	*
8179-	A0 C5	LDY	#\$C5	*
817B-	88	DEY		*
817C-	D0 FD	BNE	\$817B	*
817E-	A4 1B	LDY	\$1B	*
8180-	60	RTS		*
8181-	A2 00	LDX	#\$00	*
8183-	86 08	STX	\$08	*
8185-	A9 60	LDA	#\$60	*
8187-	85 09	STA	\$09	*
8189-	A0 27	LDY	#\$27	*
818B-	BD 2B 82	LDA	\$822B,X	*
818E-	85 06	STA	\$06	*
8190-	E8	INX		*
8191-	BD 2B 82	LDA	\$822B,X	*
8194-	85 07	STA	\$07	*
8196-	E8	INX		*
8197-	B1 06	LDA	(\$06),Y	*
8199-	91 08	STA	(\$08),Y	*
819B-	88	DEY		*
819C-	10 F9	BPL	\$8197	*
819E-	18	CLC		*
819F-	A5 08	LDA	\$08	*
81A1-	69 28	ADC	#\$28	*
81A3-	90 02	BCC	\$81A7	*
81A5-	E6 09	INC	\$09	*
81A7-	85 08	STA	\$08	*
81A9-	EC 09 03	CPX	\$0309	*
81AC-	D0 DB	BNE	\$8189	*
81AE-	60	RTS		*
81AF-	AE 09 03	LDX	\$0309	*
81B2-	A9 00	LDA	#\$00	*
81B4-	85 08	STA	\$08	*
81B6-	A9 60	LDA	#\$60	*
81B8-	85 09	STA	\$09	*
81BA-	A0 27	LDY	#\$27	*
81BC-	BD 2B 82	LDA	\$822B,X	*
81BF-	85 06	STA	\$06	*
81C1-	E8	INX		*
81C2-	BD 2B 82	LDA	\$822B,X	*
81C5-	85 07	STA	\$07	*
81C7-	E8	INX		*
81C8-	B1 08	LDA	(\$08),Y	*
81CA-	91 06	STA	(\$06),Y	*

81CC-	88	DEY		*
81CD-	10 F9	BPL	\$81C8	*
81CF-	18	CLC		*
81D0-	A5 08	LDA	\$08	*
81D2-	69 28	ADC	#\$28	*
81D4-	90 02	BCC	\$81D8	*
81D6-	E6 09	INC	\$09	*
81D8-	85 08	STA	\$08	*
81DA-	E0 00	CPX	#\$00	*
81DC-	D0 DC	BNE	\$81BA	*
81DE-	60	RTS		*
81DF-	68	PLA		*
81E0-	A8	TAY		*
81E1-	68	PLA		*
81E2-	A6 DF	LDX	\$DF	*
81E4-	9A	TXS		*
81E5-	48	PHA		*
81E6-	98	TYA		*
81E7-	48	PHA		*
81E8-	60	RTS		*
81E9-	86 1E	STX	\$1E	*
81EB-	A2 27	LDX	#\$27	*
81ED-	BD 50 06	LDA	\$0650,X	*
81F0-	9D 50 0A	STA	\$0A50,X	*
81F3-	BD D0 06	LDA	\$06D0,X	*
81F6-	9D D0 0A	STA	\$0AD0,X	*
81F9-	BD 50 07	LDA	\$0750,X	*
81FC-	9D 50 0B	STA	\$0B50,X	*
81FF-	BD D0 07	LDA	\$07D0,X	*
8202-	9D D0 0B	STA	\$0BD0,X	*
8205-	CA	DEX		*
8206-	10 E5	BPL	\$81ED	*
8208-	A6 1E	LDX	\$1E	*
820A-	60	RTS		*

APPENDIX-B

This appendix deals with 'Position Detection'. With the help of the 'DATA' file which stores the starting memory location of every row of the high resolution graphics screen, the 'ROW' and 'COL' programs search the screen and locate the boundaries of the image. 'INITIAL' and 'CENTR' subroutines calculate the centroid of the image.

A brief description of the programs with thier memory organization and program listing is given.

ASSEMBLY ROUTINE"ROW"

This routine searches the High Resolution Graphics screen row by row and accounts for the number of OFF bits in every row. It even has the capability of detecting and storing the first OFF bit in a row.

It basically consists of two main subroutines:-

1) JSR 87B2

This subroutine checks every memory location of HGR for bits which are OFF and stores the no. of OFF bits for every row in memory location \$8540-857F. It stores the position of the first OFF bit of every row in memory \$8500-853F.

2) JSR 880B

This subroutine locates the byte which contains the first OFF bit and converts it into # of bits.

MEMORY MAP

\$0-\$80	* Used for indirect access of memory from HGR
\$81	* Temporary storage for no. of rows
\$82	* Counter for incrementing the bytes in a row
\$83	* Contains \$FF to compare if all the bits are ON.
\$84	* Counter to check if it is the first OFF bit in that
	* row
\$86	* Stores the number of OFF bits in a row
\$8340-C0	* Temporary storage of Zero Page
\$8410-90	* Stores the Data File
\$8500-3F	* Stores the location of the first OFF bit in each row

\$8540-7F * Stores the no. of OFF bits in a row.

PROGRAM LISTING OF ROW

IMPORTANT REMARKS

PROGRAM LISTING OF ROW				IMPORTANT REMARKS
873F-	A2 00	LDX	#\$00	*****
8741-	A9 00	LDA	#\$00	* Initialization of memory *
8743-	9D 00 85	STA	\$8500,X	* location \$8500-8580 *
8746-	E8	INX		*****
8747-	E0 81	CPX	#\$81	* *
8749-	D0 F8	BNE	\$8743	* *
874B-	A9 01	LDA	#\$01	* *
874D-	8D 1C 03	STA	\$031C	* *
8750-	A2 00	LDX	#\$00	*****
8752-	B5 00	LDA	\$00,X	* Store ZERO PAGE in some *
8754-	9D 40 83	STA	\$8340,X	* auxiliary memory location *
8757-	E8	INX		* (\$8340-\$83BF) *
8758-	E0 90	CPX	#\$90	*****
875A-	D0 F6	BNE	\$8752	* *
875C-	A9 FF	LDA	#\$FF	* *
875E-	85 83	STA	\$83	* *
8760-	A9 07	LDA	#\$07	* *
8762-	85 89	STA	\$89	* *
8764-	A0 00	LDY	#\$00	* *
8766-	98	TYA		* *
8767-	85 81	STA	\$81	*****
8769-	A2 00	LDX	#\$00	* Initialize the counter *
876B-	8A	TXA		* for the no of rows(Y=\$81) *
876C-	85 82	STA	\$82	* and for the no of bytes *
876E-	E6 81	INC	\$81	* in a row (X=\$82) *
8770-	18	CLC		*****
8771-	B9 10 84	LDA	\$8410,Y	*****
8774-	95 00	STA	\$00,X	* Load the High Resolution *
8776-	E8	INX		* Graphics memory into the *
8777-	E8	INX		* Zero Page row by row *
8778-	6D 1C 03	ADC	\$031C	* \$8410-\$848F stores the *
877B-	E0 46	CPX	#\$46	* first memory location of *
877D-	D0 F5	BNE	\$8774	* every row of High - *
877F-	C8	INY		* Resolution Graphics *
8780-	A2 00	LDX	#\$00	*****
8782-	B9 10 84	LDA	\$8410,Y	* *
8785-	95 01	STA	\$01,X	* *
8787-	E8	INX		* *
8788-	E8	INX		* *
8789-	E0 46	CPX	#\$46	* *
878B-	D0 F8	BNE	\$8785	* *
878D-	C8	INY		* *
878E-	A9 00	LDA	#\$00	*****
8790-	85 84	STA	\$84	* Initialise some important *
8792-	85 86	STA	\$86	* counters *
8794-	A2 00	LDX	#\$00	*****
8796-	20 B2 87	JSR	\$87B2	* *
8799-	E6 82	INC	\$82	* *
879B-	E8	INX		* *

879C-	E8	INX		*	*
879D-	E0 46	CPX	#\$46	*	*
879F-	D0 F5	BNE	\$8796	*	*
87A1-	C0 80	CPY	#\$80	*	*
87A3-	D0 C4	BNE	\$8769	*	*
87A5-	A2 00	LDX	#\$00	*****	
87A7-	BD 40 83	LDA	\$8340,X	* Reload the Zero Page *	
87AA-	95 00	STA	\$00,X	* from the auxiliary memory *	
87AC-	E8	INX		*****	
87AD-	E0 90	CPX	#\$90	*	*
87AF-	D0 F6	BNE	\$87A7	*	*
87B1-	60	RTS		*****	
87B2-	A1 00	LDA	(\$00,X)	* Indirect access of HGR *	
87B4-	C5 83	CMP	\$83	* and check for the number *	
87B6-	F0 41	BEQ	\$87F9	* of bits off in the HGR *	
87B8-	86 85	STX	\$85	* byte *	
87BA-	A2 00	LDX	#\$00	*****	
87BC-	E8	INX		*	*
87BD-	E0 09	CPX	#\$09	*	*
87BF-	F0 0A	BEQ	\$87CB	*	*
87C1-	0A	ASL		*	*
87C2-	B0 F8	BCS	\$87BC	*	*
87C4-	E6 86	INC	\$86	*	*
87C6-	4C BC 87	JMP	\$87BC	*	*
87C9-	EA	NOP		*	*
87CA-	EA	NOP		*****	
87CB-	A6 84	LDX	\$84	* Searches a row for the *	
87CD-	E0 00	CPX	#\$00	* first byte with atleast *	
87CF-	D0 26	BNE	\$87F7	* one bit off and stores *	
87D1-	A9 01	LDA	#\$01	* location of this byte in *	
87D3-	85 84	STA	\$84	* the form of bits in the *	
87D5-	20 0B 88	JSR	\$880B	* memory location \$8500-3F *	
87D8-	18	CLC		*****	
87D9-	A5 87	LDA	\$87	*	*
87DB-	65 89	ADC	\$89	*	*
87DD-	85 87	STA	\$87	*	*
87DF-	38	SEC		*	*
87E0-	A5 87	LDA	\$87	*	*
87E2-	E5 86	SBC	\$86	*	*
87E4-	EA	NOP		*	*
87E5-	EA	NOP		*	*
87E6-	EA	NOP		*	*
87E7-	EA	NOP		*	*
87E8-	EA	NOP		*	*
87E9-	84 88	STY	\$88	*	*
87EB-	A4 81	LDY	\$81	*	*
87ED-	99 00 85	STA	\$8500,Y	*	*
87F0-	A4 88	LDY	\$88	*	*
87F2-	A6 85	LDX	\$85	*	*
87F4-	4C 0A 88	JMP	\$880A	*	*
87F7-	A6 85	LDX	\$85	*	*
87F9-	A9 22	LDA	#\$22	*	*

87FB-	C5 82	CMP	\$82	*	*
87FD-	D0 0B	BNE	\$880A	*	*
87FF-	84 88	STY	\$88	*	*
8801-	A4 81	LDY	\$81	*	*
8803-	A5 86	LDA	\$86	*****	
8805-	99 40 85	STA	\$8540,Y	*	* Stores the no. of OFF bits*
8808-	A4 88	LDY	\$88	*	* of a row in memory 8540-7F*
880A-	60	RTS		*	* *****
880B-	A2 00	LDX	#\$00	*	*
880D-	A5 82	LDA	\$82	*	*
880F-	85 87	STA	\$87	*	*
8811-	18	CLC		*****	
8812-	A5 87	LDA	\$87	*	* This routine converts *
8814-	65 82	ADC	\$82	*	* bytes into bits *
8816-	85 87	STA	\$87	*****	
8818-	E8	INX		*	*
8819-	E0 06	CPX	#\$06	*	*
881B-	D0 F4	BNE	\$8811	*	*
881D-	60	RTS		*	*

ASSEMBLY ROUTINE "COL"

This routine scans the High Resolution Graphics Screen vertically and registers the no. of OFF bits in every column . It scans 245 columns containing 64 bits each. It stores the no. of OFF bits of a column in memory location \$88D0-\$89C5. The heart of this M/C program is the subroutine JSR 889B. This subroutine checks for OFF bits in every column and stores them in the corressponding memory location of \$88D0-\$89C5.

MEMORY MAP

- \$00-80 * Stores one column of HGR screen at a time.
- \$83 * Increments a column in terms of bytes.
- \$84 * Increments a column in terms of bits.
- \$8340-CF * Stores the Zero Page.
- \$88D0-C5 * Stores the no. of OFF bits of the 245 columns.

PROGRAM LISTING OF COL

IMPORTANT REMARKS

8850-	A2 00	LDX	#\$00	*****
8852-	A9 00	LDA	#\$00	* Initialize \$88D0-89C5 *
8854-	9D D0 88	STA	\$88D0,X	*****
8857-	E8	INX		* *
8858-	E0 F5	CPX	#\$F5	* *
885A-	D0 F6	BNE	\$8852	* *
885C-	A2 00	LDX	#\$00	*****
885E-	B5 00	LDA	\$00,X	* Save the Zero Page in *
8860-	9D 40 83	STA	\$8340,X	* memory location \$8340- *
8863-	E8	INX		*****
8864-	E0 85	CPX	#\$85	* *
8866-	D0 F6	BNE	\$885E	* *
8868-	A9 00	LDA	#\$00	* *
886A-	85 83	STA	\$83	* *
886C-	85 84	STA	\$84	* *
886E-	A2 00	LDX	#\$00	* *
8870-	18	CLC		* *
8871-	BD 10 84	LDA	\$8410,X	*****
8874-	65 83	ADC	\$83	* Load the first byte of *
8876-	95 00	STA	\$00,X	* every row of the HGR *
8878-	E8	INX		* from the data file and *
8879-	BD 10 84	LDA	\$8410,X	* store it in Zero Page *
887C-	95 00	STA	\$00,X	*****
887E-	E8	INX		* *
887F-	E0 80	CPX	#\$80	* *
8881-	D0 ED	BNE	\$8870	* *
8883-	E6 83	INC	\$83	* *
8885-	20 9B 88	JSR	\$889B	* *
8888-	A9 23	LDA	#\$23	*****
888A-	C5 83	CMP	\$83	* Check if all the columns *
888C-	D0 E0	BNE	\$886E	* are scanned *
888E-	A2 00	LDX	#\$00	*****
8890-	BD 40 83	LDA	\$8340,X	* *
8893-	95 00	STA	\$00,X	*****
8895-	E8	INX		* Reload the Zero Page *
8896-	E0 85	CPX	#\$85	*****
8898-	D0 F6	BNE	\$8890	* *
889A-	60	RTS		* *
889B-	A0 00	LDY	#\$00	* *
889D-	A2 00	LDX	#\$00	* *
889F-	A1 00	LDA	(\$00,X)	*****
88A1-	99 00 00	STA	\$0000,Y	* Store the value of HGR *
88A4-	C8	INY		* in the Zero Page for a *
88A5-	E8	INX		* particular column *
88A6-	E8	INX		*****
88A7-	E0 80	CPX	#\$80	* *
88A9-	D0 F4	BNE	\$889F	* *
88AB-	A0 00	LDY	#\$00	* *
88AD-	A2 00	LDX	#\$00	*****

88AF-	56 00	LSR	\$00,X	* Check if the bit is OFF *
88B1-	B0 09	BCS	\$88BC	*****
88B3-	86 82	STX	\$82	* *
88B5-	A6 84	LDX	\$84	*****
88B7-	FE D0 88	INC	\$88D0,X	* Check 7 times if the bit *
88BA-	A6 82	LDX	\$82	* is OFF in a byte,if so *
88BC-	E8	INX		* increment the respective *
88BD-	E0 40	CPX	#\$40	* memory location(88D0-89C5)*
88BF-	D0 EE	BNE	\$88AF	*****
88C1-	E6 84	INC	\$84	* *
88C3-	C8	INY		* *
88C4-	C0 07	CPY	#\$07	* *
88C6-	D0 E5	BNE	\$88AD	* *
88C8-	60	RTS		* *

ASSEMBLY ROUTINE "INITIAL"

This short routine is divided into two subroutines.

1) JSR 320

It initializes the memory location which stores the result .(\$310-\$313) of "POL/MOM". Then it transfers the result of column search of the "COL" program from memory \$88D0-\$89C5 to a common memory \$9000-\$90F5. This is required by the M/C program "POL/MOM" to calculate $(X-x)**2$.

2) JSR 350

It initializes the memory location (\$310-313) which stores the result of the M/C program "POL/MOM". Then it transfers the outcome of the row search from memory \$8540-\$857F to a common memory \$9000-\$903F. This is required by "POL/MOM" to calculate $(Y-y)**2$.

PROGRAM LISTING OF INITIAL

0320-	A9 F5	LDA	#\$F5	*
0322-	8D 98 03	STA	\$0398	*
0325-	A9 00	LDA	#\$00	*
0327-	8D 10 03	STA	\$0310	*
032A-	8D 11 03	STA	\$0311	*
032D-	8D 12 03	STA	\$0312	*
0330-	8D 13 03	STA	\$0313	*
0333-	A2 00	LDX	#\$00	*
0335-	BD D0 88	LDA	\$88D0,X	*
0338-	9D 00 90	STA	\$9000,X	*
033B-	E8	INX		*
033C-	EC 98 03	CPX	\$0398	*
033F-	D0 F4	BNE	\$0335	*
0341-	60	RTS		*
0342-	EA	NOP		*
0343-	EA	NOP		*
0344-	EA	NOP		*
0345-	EA	NOP		*
0346-	EA	NOP		*
0347-	EA	NOP		*
0350-	A9 40	LDA	#\$40	*
0352-	8D 98 03	STA	\$0398	*
0355-	A2 00	LDX	#\$00	*
0357-	8E 10 03	STX	\$0310	*
035A-	8E 11 03	STX	\$0311	*
035D-	8E 12 03	STX	\$0312	*
0360-	8E 13 03	STX	\$0313	*
0363-	BD 40 85	LDA	\$8540,X	*
0366-	9D 00 90	STA	\$9000,X	*
0369-	E8	INX		*
036A-	EC 98 03	CPX	\$0398	*
036D-	D0 F4	BNE	\$0363	*
036F-	60	RTS		*

ASSEMBLY ROUTINE"CENTR"

This routine calculates the area of the image in the terms of no. of OFF bits. Then it calculates the centroid of the object.

1) Calculation of the area:

Run the M/C program "ROW" and then this part of the routine just adds the no. of OFF bits from memory location \$8540-\$857F and stores the area in \$343(LSB) and \$344(MSB).

2) Calculation of the Centroid:

Run the "COL" and "ROW" routines. Then scan the memory that stores the no. of OFF bits row by row and add them up. Stop searching as soon as the sum is equal to half the area. Store the value of the row in \$30A. this gives the y coordinate of the centroid.

Similarly search the memory that stores the no. of OFF bits column by column. Stop as soon as the sum is equal to half the area. Store the column number in \$30B, which gives the x coordinate of the centroid.

PROGRAM LISTING OF CENTR

IMPORTANT REM

```

8C00-  A2 00      LDX  #$00      *
8C02-  8E 43 03  STX  $0343     *
8C05-  8E 44 03  STX  $0344     *
8C08-  18        CLC             *
8C09-  BD 40 85  LDA  $8540,X    * Calculation of area *
8C0C-  6D 43 03  ADC  $0343     * MSB-$344; LSB-$343 *
8C0F-  8D 43 03  STA  $0343     *
8C12-  90 03     BCC  $8C17     *
8C14-  EE 44 03  INC  $0344     *
8C17-  E8        INX             *
8C18-  E0 40     CPX  #$40       *
8C1A-  D0 EC     BNE  $8C08     *
8C1C-  AD 43 03  LDA  $0343     *
8C1F-  8D 90 03  STA  $0390     * Divide area by 2 *
8C22-  AD 44 03  LDA  $0344     *
8C25-  8D 91 03  STA  $0391     * Store area/2 in 390-391 *
8C28-  18        CLC             *
8C29-  4E 90 03  LSR  $0390     *
8C2C-  6E 91 03  ROR  $0391     *
8C2F-  90 09     BCC  $8C3A     *
8C31-  18        CLC             *
8C32-  A9 80     LDA  #$80       *
8C34-  6D 90 03  ADC  $0390     *
8C37-  8D 90 03  STA  $0390     *
8C3A-  20 50 03  JSR  $0350     *
8C3D-  20 4D 8C  JSR  $8C4D     *
8C40-  8E 0B 03  STX  $030B     * Stores centroid (y) *
8C43-  20 20 03  JSR  $0320     *
8C46-  20 4D 8C  JSR  $8C4D     *
8C49-  8E 0A 03  STX  $030A     * Stores centroid(x) *
8C4C-  60        RTS             *
8C4D-  A2 00      LDX  #$00     *
8C4F-  8E 8E 03  STX  $038E     * In this subroutine *
8C52-  8E 8F 03  STX  $038F     * the particular row or *
8C55-  18        CLC             * column is identified *
8C56-  BD 00 90  LDA  $9000,X    * where the area=area/2 *
8C59-  6D 8E 03  ADC  $038E     *
8C5C-  8D 8E 03  STA  $038E     *
8C5F-  90 03     BCC  $8C64     *
8C61-  EE 8F 03  INC  $038F     *
8C64-  AD 8F 03  LDA  $038F     *
8C67-  CD 91 03  CMP  $0391     *
8C6A-  F0 04     BEQ  $8C70     *
8C6C-  E8        INX             *
8C6D-  4C 55 8C  JMP  $8C55     *
8C70-  AD 90 03  LDA  $0390     *
8C73-  8D 46 03  STA  $0346     *
8C76-  0A        ASL             *
8C77-  90 1E     BCC  $8C97     *

```

8C79-	38	SEC	*	*
8C7A-	A9 7F	LDA	#\$7F	*
8C7C-	ED 8E 03	SBC	\$038E	*
8C7F-	30 06	BMI	\$8C87	*
8C81-	20 A7 8C	JSR	\$8CA7	*
8C84-	4C 79 8C	JMP	\$8C79	*
8C87-	AD 8E 03	LDA	\$038E	*
8C8A-	29 7F	AND	#\$7F	*
8C8C-	8D 8E 03	STA	\$038E	*
8C8F-	AD 46 03	LDA	\$0346	*
8C92-	29 7F	AND	#\$7F	*
8C94-	8D 46 03	STA	\$0346	*
8C97-	38	SEC	*	*
8C98-	AD 46 03	LDA	\$0346	*
8C9B-	ED 8E 03	SBC	\$038E	*
8C9E-	30 06	BMI	\$8CA6	*
8CA0-	20 A7 8C	JSR	\$8CA7	*
8CA3-	4C 97 8C	JMP	\$8C97	*
8CA6-	60	RTS	*	*
8CA7-	E8	INX	*	*
8CA8-	18	CLC	*	*
8CA9-	BD 00 90	LDA	\$9000,X	*
8CAC-	6D 8E 03	ADC	\$038E	*
8CAF-	8D 8E 03	STA	\$038E	*
8CB2-	60	RTS	*	*

APPENDIX-C

This appendix deals with 'Orientation Dectection'. Here 'POL/MOM calculates Ixx and Iyy, whereas 'IIXXYY' computes Ixy. Ixx and Iyy are the central moments about x and y axis and Ixy is the product of inertia.

The orientation is then computed from the following geometric relationship:-

$$\tan 2A = -2*Ixy/(Ixx-Iyy)$$

where A is the angle of the major axis with respect to a row on the screen.

A brief description of the above machine language programs and thier listing is given.

ASSEMBLY ROUTINE "POL/MOM"

This machine language program is capable of calculating the second Moment of Inertia about the X-axis or about the Y-axis that is either Ixx or Iyy of the image.

Before running this routine the centroid should be loaded in memory location \$30A. and the binary file "INITIAL" should be loaded. In order to calculate Ixx JSR 320 should be executed and to calculate Iyy, JSR 350 should be executed. These subroutines will be discussed under "INITIAL".

The final values of Ixx or Iyy are stored in memory locations \$310-\$313, where \$313 is the most significant byte and \$310 is the least significant byte.

The main subroutines of the program are as follows:-

1) JSR 8A60

This subroutine calculates the absolute difference between X-x or Y-y

2) JSR 8A3D

this is a standard eight bit multiplication subroutine.

3) JSR 8AD0

This is a multiple byte Addition subroutine.

MEMORY MAP

\$301 * Stores the no. of OFF bits of a row or a column.

\$302 * Stores the MSB of $(X-x)**2$ or $(Y-y)**2$

\$303 * (LSB) Stores the LSB of the product of $N*(Y-y)**2$
* or $N*(X-x)**2$

\$304 * (MSB) where N =no of OFF bits in a row or a column

\$305 * (LSB) Stores the MSB of the above product

\$306 * (MSB) Stores the MSB of the above product.

\$30A * Stores the centroid(x or y)

\$310-13 * Stores the final result

\$9000- * Stores the no. of OFF bits of a row or column .

PROGRAM LISTING OF "POL/MQM"

IMPORTANT REM

```

89D0-  A9 00      LDA  #$00      *
89D2-  8D 99 03   STA  $0399     *
89D5-  A2 00      LDX  #$00      *
89D7-  BD 00 90   LDA  $9000,X   * Load the no of OFF bits *
89DA-  CD 99 03   CMP  $0399     *
89DD-  F0 57      BEQ  $8A36     *
89DF-  8D 01 03   STA  $0301     *
89E2-  8E 00 10   STX  $1000     *
89E5-  AD 0A 03   LDA  $030A     * Load the centroid(x or y)*
89E8-  8D 01 10   STA  $1001     *
89EB-  20 60 8A   JSR  $8A60     * Finds ABS(X-x) or (Y-y) *
89EE-  AD 03 10   LDA  $1003     *
89F1-  8D 02 10   STA  $1002     *
89F4-  20 3D 8A   JSR  $8A3D     * calculates SQR ofX-x orY-y*
89F7-  AD 00 10   LDA  $1000     *
89FA-  8D 02 03   STA  $0302     * Stores MSB of result *
89FD-  AD 01 03   LDA  $0301     *
8A00-  8D 03 10   STA  $1003     * Multiplies no of rows or*
8A03-  AD 01 10   LDA  $1001     *column by the LSB of result*
8A06-  8D 02 10   STA  $1002     *
8A09-  20 3D 8A   JSR  $8A3D     *
8A0C-  AD 01 10   LDA  $1001     *
8A0F-  8D 03 03   STA  $0303     *
8A12-  AD 00 10   LDA  $1000     * Multiplies no of rows or *
8A15-  8D 04 03   STA  $0304     *Column by MSB of the result*
8A18-  AD 02 03   LDA  $0302     *
8A1B-  8D 02 10   STA  $1002     *
8A1E-  AD 01 03   LDA  $0301     *
8A21-  8D 03 10   STA  $1003     *
8A24-  20 3D 8A   JSR  $8A3D     *
8A27-  AD 01 10   LDA  $1001     *
8A2A-  8D 05 03   STA  $0305     *
8A2D-  AD 00 10   LDA  $1000     *
8A30-  8D 06 03   STA  $0306     *
8A33-  20 D0 8A   JSR  $8AD0     * Three byte ADD subroutine *
8A36-  E8        INX             *
8A37-  EC 98 03   CPX  $0398     *
8A3A-  D0 9B      BNE  $89D7     *
8A3C-  60        RTS             *
8A3D-  A0 08      LDY  #$08     *****
8A3F-  A9 00      LDA  #$00     * 8 BIT multiplication *
8A41-  8D 00 10   STA  $1000     *
8A44-  8D 01 10   STA  $1001     *
8A47-  0A        ASL             *
8A48-  2E 00 10   ROL  $1000     *
8A4B-  0E 03 10   ASL  $1003     *
8A4E-  90 09      BCC  $8A59     *
8A50-  18        CLC             *

```

8A51-	6D 02 10	ADC	\$1002	*	*
8A54-	90 03	BCC	\$8A59	*	*
8A56-	EE 00 10	INC	\$1000	*	*
8A59-	88	DEY		*	*
8A5A-	D0 EB	BNE	\$8A47	*	*
8A5C-	8D 01 10	STA	\$1001	*	*
8A5F-	60	RTS		*****	*
8A60-	A9 FF	LDA	#\$FF	*	Absolute subtraction
8A62-	8D 02 10	STA	\$1002	*	*
8A65-	A9 00	LDA	#\$00	*	*
8A67-	8D 07 03	STA	\$0307	*	*
8A6A-	8D 08 03	STA	\$0308	*	*
8A6D-	AD 00 10	LDA	\$1000	*	*
8A70-	0A	ASL		*	*
8A71-	B0 09	BCS	\$8A7C	*	*
8A73-	AD 01 10	LDA	\$1001	*	*
8A76-	0A	ASL		*	*
8A77-	B0 0B	BCS	\$8A84	*	*
8A79-	4C 89 8A	JMP	\$8A89	*	*
8A7C-	A9 01	LDA	#\$01	*	*
8A7E-	8D 07 03	STA	\$0307	*	*
8A81-	4C 73 8A	JMP	\$8A73	*	*
8A84-	A9 01	LDA	#\$01	*	*
8A86-	8D 08 03	STA	\$0308	*	*
8A89-	AD 08 03	LDA	\$0308	*	*
8A8C-	CD 07 03	CMP	\$0307	*	*
8A8F-	F0 1D	BEQ	\$8AAE	*	*
8A91-	A9 00	LDA	#\$00	*	*
8A93-	CD 07 03	CMP	\$0307	*	*
8A96-	D0 0B	BNE	\$8AA3	*	*
8A98-	38	SEC		*	*
8A99-	AD 01 10	LDA	\$1001	*	*
8A9C-	ED 00 10	SBC	\$1000	*	*
8A9F-	8D 03 10	STA	\$1003	*	*
8AA2-	60	RTS		*	*
8AA3-	38	SEC		*	*
8AA4-	AD 00 10	LDA	\$1000	*	*
8AA7-	ED 01 10	SBC	\$1001	*	*
8AAA-	8D 03 10	STA	\$1003	*	*
8AAD-	60	RTS		*	*
8AAE-	38	SEC		*	*
8AAF-	AD 00 10	LDA	\$1000	*	*
8AB2-	ED 01 10	SBC	\$1001	*	*
8AB5-	30 04	BMI	\$8ABB	*	*
8AB7-	8D 03 10	STA	\$1003	*	*
8ABA-	60	RTS		*	*
8ABB-	ED 02 10	SBC	\$1002	*	*
8ABE-	8D 03 10	STA	\$1003	*	*
8AC1-	38	SEC		*	*
8AC2-	AD 02 10	LDA	\$1002	*	*
8AC5-	ED 03 10	SBC	\$1003	*	*
8AC8-	8D 03 10	STA	\$1003	*	*

8ACB-	EE 03 10	INC	\$1003	*	*
8ACE-	60	RTS		*****	
8ACF-	00	BRK		*	*
8ADO-	18	CLC		*	
8AD1-	AD 10 03	LDA	\$0310	*	Three byte addition
8AD4-	6D 03 03	ADC	\$0303	*	
8AD7-	8D 10 03	STA	\$0310	*	Stores the result in
8ADA-	90 03	BCC	\$8ADF	*	\$310-\$312
8ADC-	EE 11 03	INC	\$0311	*	
8ADF-	18	CLC		*	
8AE0-	AD 11 03	LDA	\$0311	*	
8AE3-	6D 04 03	ADC	\$0304	*	
8AE6-	8D 11 03	STA	\$0311	*	
8AE9-	90 03	BCC	\$8AEE	*	
8AEB-	EE 12 03	INC	\$0312	*	
8AEE-	18	CLC		*	
8AEF-	AD 11 03	LDA	\$0311	*	
8AF2-	6D 05 03	ADC	\$0305	*	
8AF5-	8D 11 03	STA	\$0311	*	
8AF8-	90 03	BCC	\$8AFD	*	
8AFA-	EE 12 03	INC	\$0312	*	
8AFD-	18	CLC		*	
8AFE-	AD 12 03	LDA	\$0312	*	
8B01-	6D 06 03	ADC	\$0306	*	
8B04-	8D 12 03	STA	\$0312	*	
8B07-	90 03	BCC	\$8B0C	*	
8B09-	EE 13 03	INC	\$0313	*	
8B0C-	60	RTS		*	

ASSEMBLY ROUTINE "IIXYY"

This program calculates the product of inertia in the given plane. Before running this routine the assembly routine 'ROW' should be executed and e loaded.

It stores the final result in memory locations(\$310-\$315),where \$310-312 store the positive values and \$313-315 store the negative values

It basically consists of 3 main subroutines:

- 1)JSR 8D87: Absolute Subtraction
- 2)JSR 8A3D: 8 BIT Multiplication
- 3)JSR 8E05: Multiple Byte Addition

MEMORY MAP

\$301- Stores the no. of OFF bits in a row
 \$303(LSB),\$304,\$305(MSB)- Temporary storage of the result
 \$30A- x coordinate of the centroid.
 \$30B y coordinate of the centroid.
 \$310 LSB of the positive result
 \$312 MSB of the positive result
 \$313 LSB of the negative result
 \$315 MSB of the negative result
 \$39D 0 if (X-x) is positive
 \$39E 0 if (Y-y) is positive
 \$396 Temporary storage for (Y-y)
 \$8500-\$853F - X coordinate of the first OFF bit in a row
 \$8540-\$857F - no. of OFF bits in a row.

\$8410-848F Table for High Resolution Memory Graphics.

PROGRAM LISTING OF IIXYY

IMPORTANT REMARKS

8D00-	A9 00	LDA	#\$00	*****
8D02-	A2 00	LDX	#\$00	* Initialize the memory *
8D04-	9D 10 03	STA	\$0310,X	* which stores the result *
8D07-	E8	INX		*****
8D08-	E0 06	CPX	#\$06	* *
8D0A-	D0 F8	BNE	\$8D04	* *
8D0C-	8D 99 03	STA	\$0399	* *
8D0F-	A0 00	LDY	#\$00	* *
8D11-	B9 40 85	LDA	\$8540,Y	* Load the no of OFF bits *
8D14-	CD 99 03	CMP	\$0399	* *
8D17-	F0 5F	BEQ	\$8D78	* *
8D19-	8D 01 03	STA	\$0301	* Store them in memory-301 *
8D1C-	8C 00 10	STY	\$1000	* *
8D1F-	AD 0B 03	LDA	\$030B	* Load the centroid(y) in 30B *
8D22-	8D 01 10	STA	\$1001	* *
8D25-	20 87 8D	JSR	\$8D87	* Find (Y-y) *
8D28-	AD 9C 03	LDA	\$039C	* *
8D2B-	8D 9D 03	STA	\$039D	* Stores 0 if positive *
8D2E-	AD 03 10	LDA	\$1003	* *
8D31-	8D 96 03	STA	\$0396	* Stores (Y-y) *
8D34-	A2 00	LDX	#\$00	* *
8D36-	B9 00 85	LDA	\$8500,Y	* Loads the first value of X *
8D39-	8D 97 03	STA	\$0397	* *
8D3C-	AD 97 03	LDA	\$0397	* *
8D3F-	8D 00 10	STA	\$1000	* *
8D42-	AD 0A 03	LDA	\$030A	* Loads the centroid(x) *
8D45-	8D 01 10	STA	\$1001	* *
8D48-	20 87 8D	JSR	\$8D87	* Finds (X-x) *
8D4B-	AD 9C 03	LDA	\$039C	* *
8D4E-	8D 9E 03	STA	\$039E	* Stores 0 if positive *
8D51-	AD 96 03	LDA	\$0396	* *
8D54-	8D 02 10	STA	\$1002	* *
8D57-	8C 1E 03	STY	\$031E	* *
8D5A-	20 3D 8A	JSR	\$8A3D	* Finds (X-x)*(Y-y) *
8D5D-	AC 1E 03	LDY	\$031E	* *
8D60-	AD 01 10	LDA	\$1001	* *
8D63-	8D 03 03	STA	\$0303	* Stores LSB of the result *
8D66-	AD 00 10	LDA	\$1000	* *
8D69-	8D 04 03	STA	\$0304	* Stores MSB of the result *
8D6C-	20 05 8E	JSR	\$8E05	* Addition Subroutine *
8D6F-	EE 97 03	INC	\$0397	* Increment value of X *
8D72-	E8	INX		* *
8D73-	EC 01 03	CPX	\$0301	* Compare if all Off bits *
8D76-	D0 C4	BNE	\$8D3C	* in that row are scanned *
8D78-	C8	INY		* *
8D79-	C0 40	CPY	#\$40	* *
8D7B-	D0 94	BNE	\$8D11	* *
8D7D-	60	RTS		* *

```

8D81- EA      NOP      *
8D82- EA      NOP      *
8D83- EA      NOP      *
8D84- EA      NOP      *
8D85- EA      NOP      *
8D86- EA      NOP      *
8D87- A9 00    LDA      #$00      * Subroutine Subtraction
8D89- 8D 07 03 STA      $0307      *
8D8C- 8D 08 03 STA      $0308      *
8D8F- AD 00 10 LDA      $1000      *
8D92- 0A      ASL      *
8D93- B0 09    BCS      $8D9E      *
8D95- AD 01 10 LDA      $1001      *
8D98- 0A      ASL      *
8D99- B0 0B    BCS      $8DA6      *
8D9B- 4C AB 8D JMP      $8DAB      *
8D9E- A9 01    LDA      #$01      *
8DA0- 8D 07 03 STA      $0307      *
8DA3- 4C 95 8D JMP      $8D95      *
8DA6- A9 01    LDA      #$01      *
8DA8- 8D 08 03 STA      $0308      *
8DAB- AD 08 03 LDA      $0308      *
8DAE- CD 07 03 CMP      $0307      *
8DB1- F0 27    BEQ      $8DDA      *
8DB3- A9 00    LDA      #$00      *
8DB5- CD 07 03 CMP      $0307      *
8DB8- D0 10    BNE      $8DCA      *
8DBA- 38      SEC      *
8DBB- AD 01 10 LDA      $1001      *
8DBE- ED 00 10 SBC      $1000      *
8DC1- 8D 03 10 STA      $1003      *
8DC4- A9 01    LDA      #$01      *
8DC6- 8D 9C 03 STA      $039C      * Stores 1 if result (-)
8DC9- 60      RTS      *
8DCA- 38      SEC      *
8DCB- AD 00 10 LDA      $1000      *
8DCE- ED 01 10 SBC      $1001      *
8DD1- 8D 03 10 STA      $1003      *
8DD4- A9 00    LDA      #$00      *
8DD6- 8D 9C 03 STA      $039C      *
8DD9- 60      RTS      *
8DDA- 38      SEC      *
8ddb- AD 00 10 LDA      $1000      *
8DDE- ED 01 10 SBC      $1001      *
8DE1- 30 09    BMI      $8DEC      *
8DE3- 8D 03 10 STA      $1003      *
8DE6- A9 00    LDA      #$00      *
8DE8- 8D 9C 03 STA      $039C      *
8DEB- 60      RTS      *
8DEC- E9 FF    SBC      #$FF      *
8DEE- 8D 03 10 STA      $1003      *
8DF1- 38      SEC      *

```

```

8DF2-  A9 FF      LDA  #$FF      *
8DF4-  ED 03 10   SBC  $1003     *
8DF7-  8D 03 10   STA  $1003     *
8DFA-  EE 03 10   INC  $1003     *
8DFD-  A9 01      LDA  #$01      *
8DFF-  8D 9C 03   STA  $039C     *
8E02-  60        RTS             *
8E03-  EA        NOP             *
8E04-  EA        NOP             *
8E05-  AD 9D 03   LDA  $039D     *****
8E08-  CD 9E 03   CMP  $039E     *   Addition Subroutine   *
8E0B-  F0 1F      BEQ  $8E2C     *   If (X-x)*(Y-y) is positive*
8E0D-  18        CLC             *   then store it in $310-312*
8E0E-  AD 13 03   LDA  $0313     *   Else in $313-$315   *
8E11-  6D 03 03   ADC  $0303     *****
8E14-  8D 13 03   STA  $0313     *
8E17-  90 03      BCC  $8E1C     *
8E19-  EE 14 03   INC  $0314     *
8E1C-  18        CLC             *
8E1D-  AD 14 03   LDA  $0314     *
8E20-  6D 04 03   ADC  $0304     *
8E23-  8D 14 03   STA  $0314     *
8E26-  90 03      BCC  $8E2B     *
8E28-  EE 15 03   INC  $0315     *
8E2B-  60        RTS             *
8E2C-  18        CLC             *
8E2D-  AD 10 03   LDA  $0310     *
8E30-  6D 03 03   ADC  $0303     *
8E33-  8D 10 03   STA  $0310     *
8E36-  90 03      BCC  $8E3B     *
8E38-  EE 11 03   INC  $0311     *
8E3B-  18        CLC             *
8E3C-  AD 11 03   LDA  $0311     *
8E3F-  6D 04 03   ADC  $0304     *
8E42-  8D 11 03   STA  $0311     *
8E45-  90 03      BCC  $8E4A     *
8E47-  EE 12 03   INC  $0312     *
8E4A-  60        RTS             *

```

APPENDIX D

This appendix deals with 'Image Recognition'. It finds nine geometric parameters which are independent of scale, position and rotation.

A uniqueness theorem(Papoulis,1965) states that if $F(x,y)$ is piecewise continuous and has nonzero values only in the finite part of the x - y plane, then moments of all orders exist and the moment sequence is uniquely determined by $F(x,y)$.

To find descriptors which are invariant in translation, rotation and size, subroutines 'CAMASM', 'ROW', 'COL', 'CENTER', 'PERI' and 'SEVEN' should be executed. These machine language routines return within 1-2 seconds the area, perimeter, and second and third central moments to the BASIC.

The BASIC routine normalizes the moments in order to compute the normalized central moments.

The normalized central moments denoted by N_{ab} , are defined as:-

$$N_{ab} = I_{ab}/(A^*L)$$

where a and b are subjected to the following constraints:

$a, b = 0$ or 1 or 2 or 3 ; and $a+b = 2$ or 3

A = area of the image and $L = a+b/2 + 1$

$I_{11} = I_{xy}$; $I_{12} = I_{xyy}$; $I_{02} = I_{yy}$; $I_{03} = I_{yyy}$;

$I_{20} = I_{xx}$; $I_{21} = I_{xxy}$; $I_{30} = I_{xxx}$;

where $I_{xx}, I_{yy}, I_{xy}, I_{xxx}, I_{yyy}, I_{xyy}, I_{xxy}$ are the second and third central moments as defined in the program description of 'SEVEN'.

From the second and third moments a set of seven parameters can be derived. They are given by:-

$$P1 = N20 + N02$$

$$P2 = (N20 - N02)^2 + 4 * N11^2$$

$$P3 = R1^2 + R2^2$$

$$P4 = R3^2 + R4^2$$

$$P5 = R1 * R3 * (R3^2 - 3 * R4^2) + R2 * R4 * (3 * R3^2 - R4^2)$$

$$P6 = (N20 - N02) * (R3^2 - R4^2) + 4 * N11 * R3 * R4$$

$$P7 = R2 * R4 * (3 * R3^2 - R4^2) - R1 * R3 * (R3^2 - 3 * R4^2)$$

This set of moments has been shown to be invariant to translation, rotation and scale change.

Two more parameters, defined as Elongation Ratio and Compactness Ratio, can be easily derived from the available information.

$$P8 = (\text{Elongation Ratio}) = I_{\max} / I_{\min}$$

where $I_{\max}$ and $I_{\min}$ are the principle moments about the major and minor axis and can be easily related to the above parameters.

$$I_{\max} = P1/2 + \text{SQRT}(P2)/2 ; \text{ and } I_{\min} = P1/2 - \text{SQRT}(P2)/2 ;$$

$$P9 = (\text{Compactness Ratio}) = (\text{perimeter})^2 / (\text{area})$$

The description and the program listing of 'SEVEN',

'CENTER' and 'PERI', which play an important role in deriving these parameters is given.

ASSEMBLY ROUTINE "CENTER"

This routine calculates the area of the image in the terms of no. of OFF bits. Then it calculates the centroid of the object.

1) Calculation of the area:

Run the M/C program "ROW" and then this part of the routine just adds the no. of OFF bits from memory location \$8540-\$857F and stores the area in \$343(LSB) and \$344(MSB).

2) Calculation of the Centroid:

Run the "COL" and "ROW" routines. Then scan the memory that stores the no. of OFF bits row by row and add them up. Stop searching as soon as the sum is equal to half the area. Store the value of the row in \$30A. this gives the y coordinate of the centroid.

Similarly search the memory that stores the no. of OFF bits column by column. Stop as soon as the sum is equal to half the area. Store the column number in \$30B, which gives the x coordinate of the centroid.

ASSEMBLY LISTING OF "CENTER"

8C00-	A2 00	LDX	#\$00	*
8C02-	8E 3E 03	STX	\$033E	*
8C05-	8E 3F 03	STX	\$033F	*
8C08-	18	CLC		*
8C09-	BD 40 85	LDA	\$8540,X	*
8C0C-	6D 3E 03	ADC	\$033E	*
8C0F-	8D 3E 03	STA	\$033E	*
8C12-	90 03	BCC	\$8C17	*
8C14-	EE 3F 03	INC	\$033F	*
8C17-	E8	INX		*
8C18-	E0 40	CPX	#\$40	*
8C1A-	D0 EC	BNE	\$8C08	*
8C1C-	AD 3E 03	LDA	\$033E	*
8C1F-	8D 90 03	STA	\$0390	*
8C22-	AD 3F 03	LDA	\$033F	*
8C25-	8D 91 03	STA	\$0391	*
8C28-	18	CLC		*
8C29-	4E 90 03	LSR	\$0390	*
8C2C-	6E 91 03	ROR	\$0391	*
8C2F-	90 09	BCC	\$8C3A	*
8C31-	18	CLC		*
8C32-	A9 80	LDA	#\$80	*
8C34-	6D 90 03	ADC	\$0390	*
8C37-	8D 90 03	STA	\$0390	*
8C3A-	20 12 87	JSR	\$8712	*
8C3D-	20 4D 8C	JSR	\$8C4D	*
8C40-	8E 0B 03	STX	\$030B	*
8C43-	20 F0 86	JSR	\$86F0	*
8C46-	20 4D 8C	JSR	\$8C4D	*
8C49-	8E 0A 03	STX	\$030A	*
8C4C	60	RTS		*
8C4D-	A2 00	LDX	#\$00	*
8C4F-	8E 8E 03	STX	\$038E	*
8C52-	8E 8F 03	STX	\$038F	*
8C55-	18	CLC		*
8C56-	BD 00 8A	LDA	\$8A00,X	*
8C59-	6D 8E 03	ADC	\$038E	*
8C5C-	8D 8E 03	STA	\$038E	*
8C5F-	90 03	BCC	\$8C64	*
8C61-	EE 8F 03	INC	\$038F	*
8C64-	AD 8F 03	LDA	\$038F	*
8C67-	CD 91 03	CMP	\$0391	*
8C6A-	F0 04	BEQ	\$8C70	*
8C6C-	E8	INX		*
8C6D-	4C 55 8C	JMP	\$8C55	*
8C70-	AD 90 03	LDA	\$0390	*
8C73-	8D 46 03	STA	\$0346	*
8C76-	0A	ASL		*

8C77-	90 1E	BCC	\$8C97	*
8C79-	38	SEC		*
8C7A-	A9 7F	LDA	#\$7F	*
8C7C-	ED 8E 03	SBC	\$038E	*
8C7F-	30 06	BMI	\$8C87	*
8C81-	20 A7 8C	JSR	\$8CA7	*
8C84-	4C 79 8C	JMP	\$8C79	*
8C87-	AD 8E 03	LDA	\$038E	*
8C8A-	29 7F	AND	#\$7F	*
8C8C-	8D 8E 03	STA	\$038E	*
8C8F-	AD 46 03	LDA	\$0346	*
8C92-	29 7F	AND	#\$7F	*
8C94-	8D 46 03	STA	\$0346	*
8C97-	38	SEC		*
8C98-	AD 46 03	LDA	\$0346	*
8C9B-	ED 8E 03	SBC	\$038E	*
8C9E-	30 06	BMI	\$8CA6	*
8CA0-	20 A7 8C	JSR	\$8CA7	*
8CA3-	4C 97 8C	JMP	\$8C97	*
8CA6-	60	RTS		*
8CA7-	E8	INX		*
8CA8-	18	CLC		*
8CA9-	BD 00 8A	LDA	\$8A00,X	*
8CAC-	6D 8E 03	ADC	\$038E	*
8CAF-	8D 8E 03	STA	\$038E	*
8CB2-	60	RTS		*

ASSEMBLY ROUTINE"PERI"

This M/C program calculates one half of the perimeter of the image. It does this in two steps.

Firstly it searches the memory \$8540-\$857F which stores the no. of OFF bits in every row. Thus it calculates the no. of rows that have atleast one OFF bit. It saves this result in \$3A1.

Then it searches the memory area(\$88D0-89C5) which store the no. of OFF bits in every column. Then it looks for the no. of columns which have atleast one OFF bit. This result is saved in \$3A2.

So \$3A1 gives the vertical span of the image while \$3A2 gives the horizontal span in terms of no. of OFF bits.

"ROW" and "COL" M/C program should be executed before running this routine.

PROGRAM LISTING OF PERI

03AE-	A2 00	LDX	#\$00	*
03B0-	A9 00	LDA	#\$00	*
03B2-	8D A1 03	STA	\$03A1	*
03B5-	8D A2 03	STA	\$03A2	*
03B8-	BD 40 85	LDA	\$8540,X	*
03BB-	F0 03	BEQ	\$03C0	*
03BD-	EE A1 03	INC	\$03A1	*
03C0-	E8	INX		*
03C1-	E0 40	CPX	#\$40	*
03C3-	D0 F3	BNE	\$03B8	*
03C5-	A2 00	LDX	#\$00	*
03C7-	BD D0 88	LDA	\$88D0,X	*
03CA-	F0 03	BEQ	\$03CF	*
03CC-	EE A2 03	INC	\$03A2	*
03CF-	E8	INX		*
03D0-	E0 F5	CPX	#\$F5	*
03D2-	D0 F3	BNE	\$03C7	*
03D4-	60	RTS		*

ASSEMBLY ROUTINE"SEVEN"

This is one of the most powerful machine language program, which is capable of calculating central moments of the image upto third order.

It can calculate the following moments:-

- | | |
|-------------------|---------|
| 1) $(X-x)^3$ | (Ixxx) |
| 2) $(Y-y)^3$ | (Iyyy) |
| 3) $(X-x)^2(Y-y)$ | (Ixxxy) |
| 4) $(X-x)(Y-y)^2$ | (Ixyy) |
| 5) $(X-x)^2$ | (Ixx) |
| 6) $(Y-y)^2$ | (Iyy) |
| 7) $(X-x)(Y-y)$ | (Ixy) |

where x and y are the coordinates of the centroid of the image.

It is essential to run "CAMASM", "ROW" and "CENTER" before executing this routine.

The main subroutines of this program are as follows:-

- 1) JSR 8C00 * Standard 8 bit Multiplication routine
- 2) JSR 8D87 * Subtraction subroutine which keeps track of
* negative result as well
- 3) JSR 8E05 * This subroutine is responsible for calculating
* all the seven parameters. It does this with
* the help of the following subroutines.
 - 3a) JSR 8E70 * Calculates the square of a number
 - 3b) JSR 8E8A * Multiple byte multiplication
 - 3c) JSR 8EC2 * Adds and stores Ixy

- 3d) JSR 8F18 * Adds and stores Iyy
- 3e) JSR 8F40 * Adds and stores Iyyy
- 3f) JSR 8FF0 * Adds and stores Ixyy
- 3g) JSR 90A0 * Adds and stores Ixxx
- 3h) JSR 9150 * Adds and stores Ixxy
- 3i) JSR 9200 * Adds and stores Ixx

MEMORY MAP

\$301 * Stores the # of OFF bits in a row.
\$303-304 * Temp. storage from 8-bit multiplication.
\$30A * Stores the x coordinate of the centroid.
\$30B * Stores the y coordinate of the centroid.
\$30E-311 * Temp. storage from JSR 8E8A
\$340-342 * Stores positive result of Ixy
\$343-345 * Stores negative result of Ixy
\$346-348 * Stores result of Iyy
\$349-34C * Stores positive result of Iyyy
\$34D-350 * Stores negative result of Iyyy
\$351-354 * Stores positive result of Ixyy
\$355-358 * Stores negative result of Ixyy
\$359-35C * Stores positive result of Ixxx
\$35C-360 * Stores negative result of Ixxx
\$361-364 * Stores positive result of Ixxy
\$365-368 * Stores negative result of Ixxy
\$369-36B * Stores result of Ixx

Note that significance of a byte for a particular storage is in the ascending order. For example in the case of Iyy the most significant byte(MSB) is \$348 and the least significant(LSB) one is \$346

PROGRAM LISTING OF SEVEN

8CC0-	A0 08	LDY	#\$08	*
8CC2-	A9 00	LDA	#\$00	*
8CC4-	8D 00 10	STA	\$1000	*
8CC7-	8D 01 10	STA	\$1001	*
8CCA-	0A	ASL		*
8CCB-	2E 00 10	ROL	\$1000	*
8CCE-	0E 03 10	ASL	\$1003	*
8CD1-	90 09	BCC	\$8CDC	*
8CD3-	18	CLC		*
8CD4-	6D 02 10	ADC	\$1002	*
8CD7-	90 03	BCC	\$8CDC	*
8CD9-	EE 00 10	INC	\$1000	*
8CDC-	88	DEY		*
8CDD-	D0 EB	BNE	\$8CCA	*
8CDF-	8D 01 10	STA	\$1001	*
8CE2-	60	RTS		*
8D00-	A9 00	LDA	#\$00	*
8D02-	A2 00	LDX	#\$00	*
8D04-	9D 40 03	STA	\$0340,X	*
8D07-	E8	INX		*
8D08-	E0 2F	CPX	#\$2F	*
8D0A-	D0 F8	BNE	\$8D04	*
8D0C-	8D 99 03	STA	\$0399	*
8D0F-	A0 00	LDY	#\$00	*
8D11-	B9 40 85	LDA	\$8540,Y	*
8D14-	CD 99 03	CMP	\$0399	*
8D17-	F0 50	BEQ	\$8D69	*
8D19-	8D 01 03	STA	\$0301	*
8D1C-	8C 00 10	STY	\$1000	*
8D1F-	AD 0B 03	LDA	\$030B	*
8D22-	8D 01 10	STA	\$1001	*
8D25-	20 87 8D	JSR	\$8D87	*
8D28-	AD 9C 03	LDA	\$039C	*
8D2B-	8D 9D 03	STA	\$039D	*
8D2E-	AD 03 10	LDA	\$1003	*
8D31-	8D 96 03	STA	\$0396	*
8D34-	A2 00	LDX	#\$00	*
8D36-	B9 00 85	LDA	\$8500,Y	*
8D39-	8D 97 03	STA	\$0397	*
8D3C-	AD 97 03	LDA	\$0397	*
8D3F-	8D 00 10	STA	\$1000	*
8D42-	AD 0A 03	LDA	\$030A	*
8D45-	8D 01 10	STA	\$1001	*
8D48-	20 87 8D	JSR	\$8D87	*
8D4B-	AD 9C 03	LDA	\$039C	*
8D4E-	8D 9E 03	STA	\$039E	*
8D51-	AD 03 10	LDA	\$1003	*
8D54-	8D 90 03	STA	\$0390	*

8D57-	8C 1E 03	STY	\$031E	*
8D5A-	20 05 8E	JSR	\$8E05	*
8D5D-	AC 1E 03	LDY	\$031E	*
8D60-	EE 97 03	INC	\$0397	*
8D63-	E8	INX		*
8D64-	EC 01 03	CPX	\$0301	*
8D67-	D0 D3	BNE	\$8D3C	*
8D69-	C8	INY		*
8D6A-	C0 40	CPY	#\$40	*
8D6C-	D0 A3	BNE	\$8D11	*
8D6E-	60	RTS		*
8D6F-	EA	NOP		*
8D70-	EA	NOP		*
8D71-	EA	NOP		*
8D72-	EA	NOP		*
8D73-	EA	NOP		*
8D74-	EA	NOP		*
8D75-	EA	NOP		*
8D76-	EA	NOP		*
8D77-	EA	NOP		*
8D78-	EA	NOP		*
8D79-	EA	NOP		*
8D7A-	EA	NOP		*
8D7B-	EA	NOP		*
8D7C-	EA	NOP		*
8D7D-	EA	NOP		*
8D7E-	EA	NOP		*
8D7F-	EA	NOP		*
8D80-	EA	NOP		*
8D81-	EA	NOP		*
8D82-	EA	NOP		*
8D83-	EA	NOP		*
8D84-	EA	NOP		*
8D85-	EA	NOP		*
8D86-	EA	NOP		*
8D87-	A9 00	LDA	#\$00	*
8D89-	8D 07 03	STA	\$0307	*
8D8C-	8D 08 03	STA	\$0308	*
8D8F-	AD 00 10	LDA	\$1000	*
8D92-	0A	ASL		*
8D93-	B0 09	BCS	\$8D9E	*
8D95-	AD 01 10	LDA	\$1001	*
8D98-	0A	ASL		*
8D99-	B0 0B	BCS	\$8DA6	*
8D9B-	4C AB 8D	JMP	\$8DAB	*
8D9E-	A9 01	LDA	#\$01	*
8DA0-	8D 07 03	STA	\$0307	*
8DA3-	4C 95 8D	JMP	\$8D95	*
8DA6-	A9 01	LDA	#\$01	*
8DA8-	8D 08 03	STA	\$0308	*
8DAB-	AD 08 03	LDA	\$0308	*
8DAE-	CD 07 03	CMP	\$0307	*

8DB1-	F0 27	BEQ	\$8DDA	*
8DB3-	A9 00	LDA	#\$00	*
8DB5-	CD 07 03	CMP	\$0307	*
8DB8-	D0 10	BNE	\$8DCA	*
8DBA-	38	SEC		*
8DBB-	AD 01 10	LDA	\$1001	*
8DBE-	ED 00 10	SBC	\$1000	*
8DC1-	8D 03 10	STA	\$1003	*
8DC4-	A9 01	LDA	#\$01	*
8DC6-	8D 9C 03	STA	\$039C	*
8DC9-	60	RTS		*
8DCA-	38	SEC		*
8DCB-	AD 00 10	LDA	\$1000	*
8DCE-	ED 01 10	SBC	\$1001	*
8DD1-	8D 03 10	STA	\$1003	*
8DD4-	A9 00	LDA	#\$00	*
8DD6-	8D 9C 03	STA	\$039C	*
8DD9-	60	RTS		*
8DDA-	38	SEC		*
8ddb-	AD 00 10	LDA	\$1000	*
8DDE-	ED 01 10	SBC	\$1001	*
8DE1-	30 09	EMI	\$8DEC	*
8DE3-	8D 03 10	STA	\$1003	*
8DE6-	A9 00	LDA	#\$00	*
8DE8-	8D 9C 03	STA	\$039C	*
8DEB-	60	RTS		*
8DEC-	E9 FF	SBC	#\$FF	*
8DEE-	8D 03 10	STA	\$1003	*
8DF1-	38	SEC		*
8DF2-	A9 FF	LDA	#\$FF	*
8DF4-	ED 03 10	SBC	\$1003	*
8DF7-	8D 03 10	STA	\$1003	*
8DFA-	EE 03 10	INC	\$1003	*
8DFD-	A9 01	LDA	#\$01	*
8DFF-	8D 9C 03	STA	\$039C	*
8E02-	60	RTS		*
8E03-	EA	NOP		*
8E04-	EA	NOP		*
8E05-	AD 96 03	LDA	\$0396	*
8E08-	8D 02 10	STA	\$1002	*
8E0B-	AD 90 03	LDA	\$0390	*
8E0E-	8D 03 10	STA	\$1003	*
8E11-	20 C0 8C	JSR	\$8CC0	*
8E14-	AD 01 10	LDA	\$1001	*
8E17-	8D 03 03	STA	\$0303	*
8E1A-	AD 00 10	LDA	\$1000	*
8E1D-	8D 04 03	STA	\$0304	*
8E20-	20 C2 8E	JSR	\$8EC2	*
8E23-	AD 96 03	LDA	\$0396	*
8E26-	8D FF 03	STA	\$03FF	*
8E29-	20 70 8E	JSR	\$8E70	*
8E2C-	20 18 8F	JSR	\$8F18	*

8E2F-	AD 96 03	LDA	\$0396	*
8E32-	8D FE 03	STA	\$03FE	*
8E35-	20 8A 8E	JSR	\$8E8A	*
8E38-	20 40 8F	JSR	\$8F40	*
8E3B-	AD 90 03	LDA	\$0390	*
8E3E-	8D FE 03	STA	\$03FE	*
8E41-	20 8A 8E	JSR	\$8E8A	*
8E44-	20 F0 8F	JSR	\$8FF0	*
8E47-	AD 90 03	LDA	\$0390	*
8E4A-	8D FF 03	STA	\$03FF	*
8E4D-	20 70 8E	JSR	\$8E70	*
8E50-	20 00 92	JSR	\$9200	*
8E53-	AD 90 03	LDA	\$0390	*
8E56-	8D FE 03	STA	\$03FE	*
8E59-	20 8A 8E	JSR	\$8E8A	*
8E5C-	20 A0 90	JSR	\$90A0	*
8E5F-	AD 96 03	LDA	\$0396	*
8E62-	8D FE 03	STA	\$03FE	*
8E65-	20 8A 8E	JSR	\$8E8A	*
8E68-	20 50 91	JSR	\$9150	*
8E6B-	60	RTS		*
8E6C-	EA	NOP		*
8E6D-	EA	NOP		*
8E6E-	EA	NOP		*
8E6F-	EA	NOP		*
8E70-	AD FF 03	LDA	\$03FF	*
8E73-	8D 02 10	STA	\$1002	*
8E76-	8D 03 10	STA	\$1003	*
8E79-	20 C0 8C	JSR	\$8CC0	*
8E7C-	AD 01 10	LDA	\$1001	*
8E7F-	8D 03 03	STA	\$0303	*
8E82-	AD 00 10	LDA	\$1000	*
8E85-	8D 04 03	STA	\$0304	*
8E88-	60	RTS		*
8E89-	EA	NOP		*
8E8A-	AD 04 03	LDA	\$0304	*
8E8D-	8D 02 10	STA	\$1002	*
8E90-	AD FE 03	LDA	\$03FE	*
8E93-	8D 03 10	STA	\$1003	*
8E96-	20 C0 8C	JSR	\$8CC0	*
8E99-	AD 00 10	LDA	\$1000	*
8E9C-	8D 11 03	STA	\$0311	*
8E9F-	AD 01 10	LDA	\$1001	*
8EA2-	8D 10 03	STA	\$0310	*
8EA5-	AD 03 03	LDA	\$0303	*
8EA8-	8D 02 10	STA	\$1002	*
8EAB-	AD FE 03	LDA	\$03FE	*
8EAE-	8D 03 10	STA	\$1003	*
8EB1-	20 C0 8C	JSR	\$8CC0	*
8EB4-	AD 00 10	LDA	\$1000	*
8EB7-	8D 0F 03	STA	\$030F	*
8EBA-	AD 01 10	LDA	\$1001	*

8EBD-	8D 0E 03	STA	\$030E	*
8ECO-	60	RTS		*
8EC1-	EA	NOP		*
8EC2-	AD 9D 03	LDA	\$039D	*
8EC5-	CD 9E 03	CMP	\$039E	*
8EC8-	D0 27	BNE	\$8EF1	*
8ECA-	18	CLC		*
8ECB-	AD 03 03	LDA	\$0303	*
8ECE-	6D 40 03	ADC	\$0340	*
8ED1-	8D 40 03	STA	\$0340	*
8ED4-	90 0B	BCC	\$8EE1	*
8ED6-	EE 41 03	INC	\$0341	*
8ED9-	AD 41 03	LDA	\$0341	*
8EDC-	D0 03	BNE	\$8EE1	*
8EDE-	EE 42 03	INC	\$0342	*
8EE1-	18	CLC		*
8EE2-	AD 04 03	LDA	\$0304	*
8EE5-	6D 41 03	ADC	\$0341	*
8EE8-	8D 41 03	STA	\$0341	*
8EEB-	90 03	BCC	\$8EF0	*
8EED-	EE 42 03	INC	\$0342	*
8EF0-	60	RTS		*
8EF1-	18	CLC		*
8EF2-	AD 03 03	LDA	\$0303	*
8EF5-	6D 43 03	ADC	\$0343	*
8EF8-	8D 43 03	STA	\$0343	*
8EFB-	90 0B	BCC	\$8F08	*
8EFD-	EE 44 03	INC	\$0344	*
8F00-	AD 44 03	LDA	\$0344	*
8F03-	D0 03	BNE	\$8F08	*
8F05-	EE 45 03	INC	\$0345	*
8F08-	18	CLC		*
8F09-	AD 04 03	LDA	\$0304	*
8F0C-	6D 44 03	ADC	\$0344	*
8F0F-	8D 44 03	STA	\$0344	*
8F12-	90 03	BCC	\$8F17	*
8F14-	EE 45 03	INC	\$0345	*
8F17-	60	RTS		*
8F18-	18	CLC		*
8F19-	AD 46 03	LDA	\$0346	*
8F1C-	6D 03 03	ADC	\$0303	*
8F1F-	8D 46 03	STA	\$0346	*
8F22-	90 0B	BCC	\$8F2F	*
8F24-	EE 47 03	INC	\$0347	*
8F27-	AD 47 03	LDA	\$0347	*
8F2A-	D0 03	BNE	\$8F2F	*
8F2C-	EE 48 03	INC	\$0348	*
8F2F-	18	CLC		*
8F30-	AD 47 03	LDA	\$0347	*
8F33-	6D 04 03	ADC	\$0304	*
8F36-	8D 47 03	STA	\$0347	*
8F39-	90 03	BCC	\$8F3E	*

8F3B-	EE 48 03	INC	\$0348	*
8F3E-	60	RTS		*
8F3F-	EA	NOP		*
8F40-	AD 9D 03	LDA	\$039D	*
8F43-	DO 55	BNE	\$8F9A	*
8F45-	18	CLC		*
8F46-	AD 0E 03	LDA	\$030E	*
8F49-	6D 49 03	ADC	\$0349	*
8F4C-	8D 49 03	STA	\$0349	*
8F4F-	90 0B	BCC	\$8F5C	*
8F51-	EE 4A 03	INC	\$034A	*
8F54-	AD 4A 03	LDA	\$034A	*
8F57-	DO 03	BNE	\$8F5C	*
8F59-	EE 4B 03	INC	\$034B	*
8F5C-	18	CLC		*
8F5D-	AD 0F 03	LDA	\$030F	*
8F60-	6D 4A 03	ADC	\$034A	*
8F63-	8D 4A 03	STA	\$034A	*
8F66-	90 0B	BCC	\$8F73	*
8F68-	EE 4B 03	INC	\$034B	*
8F6B-	AD 4B 03	LDA	\$034B	*
8F6E-	DO 03	BNE	\$8F73	*
8F70-	EE 4C 03	INC	\$034C	*
8F73-	18	CLC		*
8F74-	AD 10 03	LDA	\$0310	*
8F77-	6D 4A 03	ADC	\$034A	*
8F7A-	8D 4A 03	STA	\$034A	*
8F7D-	90 0B	BCC	\$8F8A	*
8F7F-	EE 4B 03	INC	\$034B	*
8F82-	AD 4B 03	LDA	\$034B	*
8F85-	DO 03	BNE	\$8F8A	*
8F87-	EE 4C 03	INC	\$034C	*
8F8A-	18	CLC		*
8F8B-	AD 11 03	LDA	\$0311	*
8F8E-	6D 4B 03	ADC	\$034B	*
8F91-	8D 4B 03	STA	\$034B	*
8F94-	90 03	BCC	\$8F99	*
8F96-	EE 4C 03	INC	\$034C	*
8F99-	60	RTS		*
8F9A-	18	CLC		*
8F9B-	AD 0E 03	LDA	\$030E	*
8F9E-	6D 4D 03	ADC	\$034D	*
8FA1-	8D 4D 03	STA	\$034D	*
8FA4-	90 0B	BCC	\$8FB1	*
8FA6-	EE 4E 03	INC	\$034E	*
8FA9-	AD 4E 03	LDA	\$034E	*
8FAC-	DO 03	BNE	\$8FB1	*
8FAE-	EE 4F 03	INC	\$034F	*
8FB1-	18	CLC		*
8FB2-	AD 0F 03	LDA	\$030F	*
8FB5-	6D 4E 03	ADC	\$034E	*
8FB8-	8D 4E 03	STA	\$034E	*

8FBB-	90 0B	BCC	\$8FC8	*
8FBD-	EE 4F 03	INC	\$034F	*
8FC0-	AD 4F 03	LDA	\$034F	*
8FC3-	D0 03	BNE	\$8FC8	*
8FC5-	EE 50 03	INC	\$0350	*
8FC8-	18	CLC		*
8FC9-	AD 10 03	LDA	\$0310	*
8FCC-	6D 4E 03	ADC	\$034E	*
8FCF-	8D 4E 03	STA	\$034E	*
8FD2-	90 0B	BCC	\$8FDF	*
8FD4-	EE 4F 03	INC	\$034F	*
8FD7-	AD 4F 03	LDA	\$034F	*
8FDA-	D0 03	BNE	\$8FDF	*
8FDC-	EE 50 03	INC	\$0350	*
8FDF-	18	CLC		*
8FE0-	AD 11 03	LDA	\$0311	*
8FE3-	6D 4F 03	ADC	\$034F	*
8FE6-	8D 4F 03	STA	\$034F	*
8FE9-	90 03	BCC	\$8FEE	*
8FEB-	EE 50 03	INC	\$0350	*
8FEE-	60	RTS		*
8FEF-	EA	NOP		*
8FF0-	AD 9E 03	LDA	\$039E	*
8FF3-	D0 55	BNE	\$904A	*
8FF5-	18	CLC		*
8FF6-	AD 0E 03	LDA	\$030E	*
8FF9-	6D 51 03	ADC	\$0351	*
8FFC-	8D 51 03	STA	\$0351	*
8FFF-	90 0B	BCC	\$900C	*
9001-	EE 52 03	INC	\$0352	*
9004-	AD 52 03	LDA	\$0352	*
9007-	D0 03	BNE	\$900C	*
9009-	EE 53 03	INC	\$0353	*
900C-	18	CLC		*
900D-	AD 0F 03	LDA	\$030F	*
9010-	6D 52 03	ADC	\$0352	*
9013-	8D 52 03	STA	\$0352	*
9016-	90 0B	BCC	\$9023	*
9018-	EE 53 03	INC	\$0353	*
901B-	AD 53 03	LDA	\$0353	*
901E-	D0 03	BNE	\$9023	*
9020-	EE 54 03	INC	\$0354	*
9023-	18	CLC		*
9024-	AD 10 03	LDA	\$0310	*
9027-	6D 52 03	ADC	\$0352	*
902A-	8D 52 03	STA	\$0352	*
902D-	90 0B	BCC	\$903A	*
902F-	EE 53 03	INC	\$0353	*
9032-	AD 53 03	LDA	\$0353	*
9035-	D0 03	BNE	\$903A	*
9037-	EE 54 03	INC	\$0354	*
903A-	18	CLC		*

903B-	AD 11 03	LDA	\$0311	*
903E-	6D 53 03	ADC	\$0353	*
9041-	8D 53 03	STA	\$0353	*
9044-	90 03	BCC	\$9049	*
9046-	EE 54 03	INC	\$0354	*
9049-	60	RTS		*
904A-	18	CLC		*
904B-	AD 0E 03	LDA	\$030E	*
904E-	6D 55 03	ADC	\$0355	*
9051-	8D 55 03	STA	\$0355	*
9054-	90 0B	BCC	\$9061	*
9056-	EE 56 03	INC	\$0356	*
9059-	AD 56 03	LDA	\$0356	*
905C-	D0 03	BNE	\$9061	*
905E-	EE 57 03	INC	\$0357	*
9061-	18	CLC		*
9062-	AD 0F 03	LDA	\$030F	*
9065-	6D 56 03	ADC	\$0356	*
9068-	8D 56 03	STA	\$0356	*
906B-	90 0B	BCC	\$9078	*
906D-	EE 57 03	INC	\$0357	*
9070-	AD 57 03	LDA	\$0357	*
9073-	D0 03	BNE	\$9078	*
9075-	EE 58 03	INC	\$0358	*
9078-	18	CLC		*
9079-	AD 10 03	LDA	\$0310	*
907C-	6D 56 03	ADC	\$0356	*
907F-	8D 56 03	STA	\$0356	*
9082-	90 0B	BCC	\$908F	*
9084-	EE 57 03	INC	\$0357	*
9087-	AD 57 03	LDA	\$0357	*
908A-	D0 03	BNE	\$908F	*
908C-	EE 58 03	INC	\$0358	*
908F-	18	CLC		*
9090-	AD 11 03	LDA	\$0311	*
9093-	6D 57 03	ADC	\$0357	*
9096-	8D 57 03	STA	\$0357	*
9099-	90 03	BCC	\$909E	*
909B-	EE 58 03	INC	\$0358	*
909E-	60	RTS		*
909F-	EA	NOP		*
90A0-	AD 9E 03	LDA	\$039E	*
90A3-	D0 55	BNE	\$90FA	*
90A5-	18	CLC		*
90A6-	AD 0E 03	LDA	\$030E	*
90A9-	6D 59 03	ADC	\$0359	*
90AC-	8D 59 03	STA	\$0359	*
90AF-	90 0B	BCC	\$90BC	*
90B1-	EE 5A 03	INC	\$035A	*
90B4-	AD 5A 03	LDA	\$035A	*
90B7-	D0 03	BNE	\$90BC	*
90B9-	EE 5B 03	INC	\$035B	*

90BC-	18	CLC		*
90BD-	AD 0F 03	LDA	\$030F	*
90C0-	6D 5A 03	ADC	\$035A	*
90C3-	8D 5A 03	STA	\$035A	*
90C6-	90 0B	BCC	\$90D3	*
90C8-	EE 5B 03	INC	\$035B	*
90CB-	AD 5B 03	LDA	\$035B	*
90CE-	D0 03	BNE	\$90D3	*
90D0-	EE 5C 03	INC	\$035C	*
90D3-	18	CLC		*
90D4-	AD 10 03	LDA	\$0310	*
90D7-	6D 5A 03	ADC	\$035A	*
90DA-	8D 5A 03	STA	\$035A	*
90DD-	90 0B	BCC	\$90EA	*
90DF-	EE 5B 03	INC	\$035B	*
90E2-	AD 5B 03	LDA	\$035B	*
90E5-	D0 03	BNE	\$90EA	*
90E7-	EE 5C 03	INC	\$035C	*
90EA-	18	CLC		*
90EB-	AD 11 03	LDA	\$0311	*
90EE-	6D 5B 03	ADC	\$035B	*
90F1-	8D 5B 03	STA	\$035B	*
90F4-	90 03	BCC	\$90F9	*
90F6-	EE 5C 03	INC	\$035C	*
90F9-	60	RTS		*
90FA-	18	CLC		*
90FB-	AD 0E 03	LDA	\$030E	*
90FE-	6D 5D 03	ADC	\$035D	*
9101-	8D 5D 03	STA	\$035D	*
9104-	90 0B	BCC	\$9111	*
9106-	EE 5E 03	INC	\$035E	*
9109-	AD 5E 03	LDA	\$035E	*
910C-	D0 03	BNE	\$9111	*
910E-	EE 5F 03	INC	\$035F	*
9111-	18	CLC		*
9112-	AD 0F 03	LDA	\$030F	*
9115-	6D 5E 03	ADC	\$035E	*
9118-	8D 5E 03	STA	\$035E	*
911B-	90 0B	BCC	\$9128	*
911D-	EE 5F 03	INC	\$035F	*
9120-	AD 5F 03	LDA	\$035F	*
9123-	D0 03	BNE	\$9128	*
9125-	EE 60 03	INC	\$0360	*
9128-	18	CLC		*
9129-	AD 10 03	LDA	\$0310	*
912C-	6D 5E 03	ADC	\$035E	*
912F-	8D 5E 03	STA	\$035E	*
9132-	90 0B	BCC	\$913F	*
9134-	EE 5F 03	INC	\$035F	*
9137-	AD 5F 03	LDA	\$035F	*
913A-	D0 03	BNE	\$913F	*
913C-	EE 60 03	INC	\$0360	*

913F-	18	CLC		*
9140-	AD 11 03	LDA	\$0311	*
9143-	6D 5F 03	ADC	\$035F	*
9146-	8D 5F 03	STA	\$035F	*
9149-	90 03	BCC	\$914E	*
914B-	EE 60 03	INC	\$0360	*
914E-	60	RTS		*
914F-	EA	NOP		*
9150-	AD 9D 03	LDA	\$039D	*
9153-	D0 55	BNE	\$91AA	*
9155-	18	CLC		*
9156-	AD 0E 03	LDA	\$030E	*
9159-	6D 61 03	ADC	\$0361	*
915C-	8D 61 03	STA	\$0361	*
915F-	90 0B	BCC	\$916C	*
9161-	EE 62 03	INC	\$0362	*
9164-	AD 62 03	LDA	\$0362	*
9167-	D0 03	BNE	\$916C	*
9169-	EE 63 03	INC	\$0363	*
916C-	18	CLC		*
916D-	AD 0F 03	LDA	\$030F	*
9170-	6D 62 03	ADC	\$0362	*
9173-	8D 62 03	STA	\$0362	*
9176-	90 0B	BCC	\$9183	*
9178-	EE 63 03	INC	\$0363	*
917B-	AD 63 03	LDA	\$0363	*
917E-	D0 03	BNE	\$9183	*
9180-	EE 64 03	INC	\$0364	*
9183-	18	CLC		*
9184-	AD 10 03	LDA	\$0310	*
9187-	6D 62 03	ADC	\$0362	*
918A-	8D 62 03	STA	\$0362	*
918D-	90 0B	BCC	\$919A	*
918F-	EE 63 03	INC	\$0363	*
9192-	AD 63 03	LDA	\$0363	*
9195-	D0 03	BNE	\$919A	*
9197-	EE 64 03	INC	\$0364	*
919A-	18	CLC		*
919B-	AD 11 03	LDA	\$0311	*
919E-	6D 63 03	ADC	\$0363	*
91A1-	8D 63 03	STA	\$0363	*
91A4-	90 03	BCC	\$91A9	*
91A6-	EE 64 03	INC	\$0364	*
91A9-	60	RTS		*
91AA-	18	CLC		*
91AB-	AD 0E 03	LDA	\$030E	*
91AE-	6D 65 03	ADC	\$0365	*
91B1-	8D 65 03	STA	\$0365	*
91B4-	90 0B	BCC	\$91C1	*
91B6-	EE 66 03	INC	\$0366	*
91B9-	AD 66 03	LDA	\$0366	*
91BC-	D0 03	BNE	\$91C1	*

91BE-	EE 67 03	INC	\$0367	*
91C1-	18	CLC		*
91C2-	AD 0F 03	LDA	\$030F	*
91C5-	6D 66 03	ADC	\$0366	*
91C8-	8D 66 03	STA	\$0366	*
91CB-	90 0B	BCC	\$91D8	*
91CD-	EE 67 03	INC	\$0367	*
91D0-	AD 67 03	LDA	\$0367	*
91D3-	D0 03	BNE	\$91D8	*
91D5-	EE 68 03	INC	\$0368	*
91D8-	18	CLC		*
91D9-	AD 10 03	LDA	\$0310	*
91DC-	6D 66 03	ADC	\$0366	*
91DF-	8D 66 03	STA	\$0366	*
91E2-	90 0B	BCC	\$91EF	*
91E4-	EE 67 03	INC	\$0367	*
91E7-	AD 67 03	LDA	\$0367	*
91EA-	D0 03	BNE	\$91EF	*
91EC-	EE 68 03	INC	\$0368	*
91EF-	18	CLC		*
91F0-	AD 11 03	LDA	\$0311	*
91F3-	6D 67 03	ADC	\$0367	*
91F6-	8D 67 03	STA	\$0367	*
91F9-	90 03	BCC	\$91FE	*
91FB-	EE 68 03	INC	\$0368	*
91FE-	60	RTS		*
91FF-	EA	NOP		*
9200-	18	CLC		*
9201-	AD 69 03	LDA	\$0369	*
9204-	6D 03 03	ADC	\$0303	*
9207-	8D 69 03	STA	\$0369	*
920A-	90 0B	BCC	\$9217	*
920C-	EE 6A 03	INC	\$036A	*
920F-	AD 6A 03	LDA	\$036A	*
9212-	D0 03	BNE	\$9217	*
9214-	EE 6B 03	INC	\$036B	*
9217-	18	CLC		*
9218-	AD 6A 03	LDA	\$036A	*
921B-	6D 04 03	ADC	\$0304	*
921E-	8D 6A 03	STA	\$036A	*
9221-	90 03	BCC	\$9226	*
9223-	EE 6B 03	INC	\$036B	*
9226	60	RTS		*

MAIN PROGRAM LISTING

```

1 D$ = ""
2 PRINT D$;"BLOAD ROW,D2"
3 PRINT D$;"BLOAD COL"
4 PRINT D$;"BLOAD CENTER"
5 PRINT D$;"BLOAD ROW+INT"
6 PRINT D$;"BLOAD DATA,A$8410,D1"
7 PRINT D$;"BLOAD CAMASM"
8 GOSUB 6000
9 DIM X1(4),Y1(4),E(4),S8(4),P2(4),UU(7,10)
12 DIM M1(10),M2(10),M3(10),M4(10),M5(10),M6(10),M7(10)
15 REM ***** MAIN MENU*****
17 HOME : VTAB 3: HTAB 4: PRINT "WHAT WOULD YOU LIKE TO DO?"
19 VTAB 6: HTAB 7: PRINT "(1) SETUP THE SYSTEM "
21 HTAB 7: PRINT "(2) UPDATE THE SYSTEM "
23 HTAB 7: PRINT "(3) RUN THE SYSTEM "
25 HTAB 7: PRINT "(4) SETUP THE CALIBRATION"
27 VTAB 3: HTAB 31: GET I$:S = VAL (I$)
28 PRINT S
29 IF S < 1 OR S > 4 THEN GOTO 27
31 HOME : IF S = 1 THEN GOSUB 3000
33 IF S = 2 THEN GOSUB 3500
35 IF S = 3 THEN GOSUB 4000
37 IF S = 4 THEN GOSUB 9000
39 VTAB 20: HTAB 5: PRINT " PRESS'R'(TO RETURN TO MAIN MENU)"
41 INPUT X$: IF X$ = "R" THEN GOSUB 6000: GOTO 15
43 END
90 REM ***ROBOT IN ACTION*****
100 H = 7.69:L = 7.0:LL = 3.5:PI = 3.14159
110 C = 180 / PI:R1 = 1
120 S1 = 1125:S2 = S1:S3 = 672:S4 = 241:S5 = S4:S6 = 306
130 U = 1
131 FOR IO = 1 TO OB:E(IO) = ABS (P2(IO) - SUM) / SUM: NEXT IO
132 LEAST = E(1):IJ = 1
133 IF OB = 1 THEN GOTO 139
134 FOR IO = 2 TO OB:
135 IF E(IO) < LEAST THEN LEAST = E(IO):IJ = IO
136 NEXT IO
139 D$ = ""
140 PRINT D$;"PR#2"
150 PRINT "@RESET"
155 IP = 1
160 X = M1(IP):Y = M2(IP):Z = M3(IP):P = M4(IP):R = M5(IP):JAW = M6(IP):S
    = M7(IP):IP = IP + 1
180 GOSUB 1000
190 W1 = INT (S1 * T1):W2 = INT (S2 * T2):W3 = INT (S3 * T3):W4 = INT
    (S4 * T4):W5 = INT (S5 * T5)
200 X = M1(IP):Y = M2(IP):Z = M3(IP):P = M4(IP):R = M5(IP):JAW = M6(IP):S
    = M7(IP):IP = IP + 1
205 IF X = 100 THEN X = F1:Y = F2
207 IF X = 200 THEN X = X1(IJ):Y = Y1(IJ):S = S1(IJ)

```

```

208 IF JAW = 600 THEN JAW = JO(IJ) + 200
210 IF X < - 100 GOTO 650
220 GOSUB 1000
230 C1 = INT (S1 * T1) - W1
240 C2 = INT (S2 * T2) - W2:C3 = INT (S3 * T3) - W3:C4 = INT (S4 * T4)
   - W4:C5 = INT (S5 * T5) - W5
250 W1 = W1 + C1:W2 = W2 + C2:W3 = W3 + C3:W4 = W4 + C4:W5 = W5 + C5
260 PRINT "@STEP",S,C1,- C2,C3,- C4,C5
265 U = U + 1:UU(1,U) = C1:UU(2,U) = - C2:UU(3,U) = C3:UU(4,U) = - C4:U
   U(5,U) = C5:UU(6,U) = JAW:UU(7,U) = S
270 IF JAW < 0 THEN GOTO 300
275 PRINT "@STEP",S,0,0,0,0,0,JAW
280 GOTO 200
300 PRINT "@CLOSE",160
310 PRINT "@STEP",S,0,0,0,0,0,JAW
320 GOTO 200
500 FOR I = 2 TO U
510 PRINT "@STEP",UU(7,I),UU(1,I),UU(2,I),UU(3,I),UU(4,I),UU(5,I)
520 IF UU(6,I) < 0 GOTO 560
530 PRINT "@STEP",UU(7,I),0,0,0,0,0,UU(6,I)
540 GOTO 600
560 PRINT "@CLOSE",160
570 PRINT "@STEP",UU(7,I),0,0,0,0,0,UU(6,I)
600 NEXT I
650 HOME : PRINT "DO YOU WANT TO REPEAT THE SEQUENCE ?"
660 INPUT L$
670 IF L$ = "YES" THEN GOTO 500
680 RETURN
700 END
705 PRINT "AS SOON AS THE OBJECT IS IN THE CAMERA'S VIEW INPUT 0": INPUT
   0
707 HGR2 : TEXT
710 CALL 32768
715 CALL 34896
720 CALL 34623
725 CALL 35840
740 SUM = PEEK (830) + 256 * PEEK (831)
750 PRINT "AREA =",SUM
774 REM *****ROUTINE FOR CENTROID CALC**
810 X2 = PEEK (778):Y2 = PEEK (779)
845 PRINT D$;"OPEN TRANS"
850 PRINT D$;"READ TRANS"
855 INPUT A1: INPUT B1: INPUT A2: INPUT B2
860 PRINT D$;"CLOSE TRANS"
875 PRINT X2,Y2
880 F1 = B1 + A1 * X2
885 F2 = B2 + Y2 / 50
886 IF A2 = 1 THEN F2 = B2 - Y2 / 50
890 PRINT F1,F2
895 RETURN
1000 P = P / C:R = R / C
1010 RR = SQR (X * X + Y * Y)
1020 IF RR < 2.25 THEN 1600

```

```

1030 IF X = 0 THEN 1055
1040 T1 = ATN (Y / X): GOTO 1060
1055 T1 = SGN (Y) * PI / 2
1060 IF X < 0 GOTO 1600
1070 T4 = P + R + R1 * T1
1080 T5 = P - R - R1 * T1
1090 IF T4 < - 270 / C OR T4 > 270 / C THEN GOTO 1600
1100 IF T5 < - 270 / C OR T5 > 270 / C THEN GOTO 1600
1110 RO = RR - LL * COS (P)
1120 ZO = Z - LL * SIN (P) - H
1130 IF RO < 2.25 GOTO 1600
1140 IF RO = 0 THEN GOTO 1160
1150 G = ATN (ZO / RO): GOTO 1162
1160 G = SGN (ZO) * PI / 2
1162 A = RO * RO + ZO * ZO
1164 A = 4 * L * L / A - 1
1170 IF A < 0 THEN GOTO 1600
1180 A = ATN ( SQR (A))
1190 T2 = A + G
1200 T3 = G - A
1210 IF T2 > 144 / C OR T2 < - 127 / C GOTO 1600
1220 IF (T2 - T3) < 0 OR (T2 - T3) > 149 / C GOTO 1600
1230 RETURN
1600 PRINT "***** OUT OF REACH*****"
1610 PRINT " AS THE X-Y COORDINATES OF THE DESTINATION ARE OUT OF REACH
YOU WILL BE RETURNED TO MAIN MENU
1620 FOR I = 1 TO 100: NEXT I
1630 GOTO 15
3000 REM ***SET UP ROUTINE***
3010 VTAB 2: HTAB 3: PRINT " YOU WANT TO CHOOSE FROM HOW MANY OBJECTS ?"

3015 INPUT OB:
3020 PRINT D$;"OPEN SETUP"
3025 PRINT D$;"DELETE SETUP"
3030 PRINT D$;"OPEN SETUP"
3035 PRINT D$;"WRITE SETUP"
3040 PRINT OB
3045 PRINT D$;"CLOSE SETUP"
3050 FOR IO = 1 TO OB: GOSUB 3100: NEXT IO
3060 RETURN
3100 REM ***** STORE DATA IN TEXTFILE***
3105 HOME : PRINT "PLACE THE OBJECT NO",IO,"IN THE VIEW OF THE CAMERA"
3110 GOSUB 705
3115 PRINT "INPUT THE X-Y COORDINATES OF THE DESTINATION OF THE OBJECT"
3120 INPUT X1(IO),Y1(IO)
3125 PRINT "INPUT THE SPEED"
3130 INPUT S8(IO)
3135 JO(IO) = 200 + SUM / 6
3136 IF JO(IO) > 600 THEN JO(IO) = 600
3140 PRINT D$;"APPEND SETUP"
3145 PRINT D$;"WRITE SETUP"
3150 PRINT X1(IO): PRINT Y1(IO): PRINT JO(IO): PRINT S8(IO): PRINT SUM
3155 PRINT D$;"CLOSE SETUP"
3160 RETURN

```

```

3500 REM *****UPDATING*****
3505 PRINT "HOW MANY OBJECTS YOU WANT TO ADD TO THE SYSTEM ?"
3507 INPUT ADD
3510 PRINT D$;"OPEN SETUP"
3515 PRINT D$;"POSITION SETUP,RO"
3520 PRINT D$;"READ SETUP"
3525 INPUT OB
3530 PRINT D$;"CLOSE SETUP"
3535 OS = OB + ADD
3540 PRINT D$;"OPEN SETUP"
3545 PRINT D$;"POSITION SETUP,RO"
3550 PRINT D$;"WRITE SETUP"
3555 PRINT OS
3560 PRINT D$;"CLOSE SETUP"
3565 FOR IO = OB + 1 TO OS: GOSUB 3100: NEXT IO
3570 RETURN
4000 REM ***** EXECUTION*****
4010 PRINT D$;"OPEN SETUP"
4020 PRINT D$;"READ SETUP"
4025 INPUT OB
4030 FOR IO = 1 TO OB: INPUT X1(IO): INPUT Y1(IO): INPUT JO(IO): INPUT S
8(IO): INPUT P2(IO): NEXT IO
4031 PRINT D$;"CLOSE SETUP"
4032 IF HO = 0 THEN GOSUB 7500
4035 GOSUB 705
4045 GOSUB 90
4050 RETURN
5600 REM *** CALC PARMS ***
5610 LET RC = AR * (3 - SE) * 128
5612 RH = INT (RC / 256): RL = RC - (RH * 256)
5614 POKE 777,RL: POKE 778,RH
5630 LET CB = INT (PW * (3 - SE) * 128 / 7) + 1
5640 IF CB = 74 THEN CB = 73
5650 LET CU = CB
5660 IF CU > 40 THEN CU = 40
5670 POKE 776,CB
5680 POKE 780,CU
5690 LET TB = CB * RC / 2
5692 LET EX = 0
5700 LET ST = 100 * TB / BA
5705 EP = (256 * CU) + (RC / 2): REM EPSON PARMS
5710 LET CM = 192
5720 IF SE = 1 THEN CM = CM + 32
5730 IF PW = 1 THEN CM = CM + 16
5740 IF AR = 2 THEN CM = CM + 4
5750 LET EX = 1
5760 IF ( NOT FE) OR RE OR ST > ET THEN SK = ET: GOTO 5810
5770 IF ET = 0 THEN SK = 0: GOTO 5790
5780 LET SK = ET - ST
5800 LET CM = CM + 2
5810 LET I = INT (SK / 256): POKE 770,I
5820 LET I = SK - (I * 256): POKE 769,I + 100

```



```

5825 POKE 768,CM: POKE 779,EX
5827 POKE 779,0
5830 RETURN
6000 REM *** RECALL SETUP ***
6010 ONERR GOTO 6100
6015 D$ = ""
6020 PRINT D$;"OPEN EYEPARMS"
6030 PRINT D$;"READ EYEPARMS"
6040 INPUT ET: INPUT BA: INPUT SL
6041 INPUT AR: INPUT SE: INPUT PW
6042 INPUT LL: INPUT LM: INPUT FE
6043 INPUT RE
6050 PRINT D$;"CLOSE EYEPARMS"
6060 LET SU = 1
6070 POKE 771,16 * SL
6080 LET FT = 0
6090 GOSUB 5600: REM CALCPARMS
6092 POKE 33792,200: POKE 216,0
6095 RETURN
6999 D$ = ""
7000 PRINT D$;"OPEN X-Y"
7010 PRINT D$;"DELETE X-Y"
7020 PRINT D$;"OPEN X-Y"
7021 II = 1: DIM G1(20),G2(20),G3(20),G4(20),G5(20),G6(20),G7(20)
7022 PRINT "INPUT THE ",II,"POSITION OF THE ROBOT"
7024 PRINT "INPUT 0 FOR THE X AFTER THE LAST POSITION"
7026 PRINT "INPUT X,Y,Z,PITCH,ROLL,JAW,SPEED"
7028 INPUT G1(II),G2(II),G3(II),G4(II),G5(II),G6(II),G7(II)
7029 IF G1(II) = 0 THEN GOTO 7090
7031 II = II + 1
7032 GOTO 7022
7090 PRINT D$;"WRITE X-Y"
7092 IU = II - 1: PRINT IU
7095 FOR IX = 1 TO II - 1: PRINT G1(IX): PRINT G2(IX): PRINT G3(IX): PRINT
G4(IX): PRINT G5(IX): PRINT G6(IX): PRINT G7(IX)
7100 NEXT IX
7110 PRINT D$;"CLOSE X-Y"
7120 END
7500 REM LOAD THE COORDINATE DATA
7510 PRINT D$;"OPEN X-Y"
7520 PRINT D$;"READ X-Y"
7530 INPUT IU
7540 FOR I8 = 1 TO IU: INPUT M1(I8): INPUT M2(I8): INPUT M3(I8): INPUT M
4(I8): INPUT M5(I8): INPUT M6(I8): INPUT M7(I8): NEXT I8
7550 PRINT D$;"CLOSE X-Y"
7555 HO = 1
7560 RETURN
9000 PRINT "INPUT THE X-Y COORDINATES OF THE TABLE"
9003 INPUT T1,T3
9004 PRINT T3
9005 CALL 32768: CALL 34896
9006 CALL 34623: CALL 35840
9007 S1 = PEEK (778):S3 = PEEK (779)

```

```
9008 PRINT "INPUT THE X&Y COORDINATES OF THE TABLE"
9010 INPUT T2,T4
9012 CALL 32768: CALL 34896
9013 CALL 34623: CALL 35840
9014 S2 = PEEK (778):S4 = PEEK (779): HGR2 : TEXT
9015 PRINT S1: PRINT S2: PRINT S3: PRINT S4
9020 A1 = (T1 - T2) / (S1 - S2):B1 = T1 - A1 * S1
9023 A2 = 1
9025 IF (S3 - S4) > 0 AND (T3 - T4) > 0 THEN A2 = 0
9030 B2 = T3 + S3 / 50
9031 PRINT A1,B1,A2,B2,T3,S3
9032 PRINT D$;"OPEN TRANS"
9033 PRINT D$;"DELETE TRANS"
9035 PRINT D$;"OPEN TRANS"
9037 PRINT D$;"WRITE TRANS"
9038 PRINT A1: PRINT B1: PRINT A2: PRINT B2
9039 PRINT D$;"CLOSE TRANS"
9040 RETURN
```